The Dissertation Committee for Chalongrath Pholsiri certifies that this is the approved version of the following dissertation:

TASK-BASED DECISION MAKING AND CONTROL OF ROBOTIC MANIPULATORS

Committee:

Delbert Tesar, Supervisor

Chetan Kapoor, Co-Supervisor

Benito Fernandez

Richard Crawford

Benjamin Kuipers

Richard Hooper

TASK-BASED DECISION MAKING AND CONTROL OF ROBOTIC MANIPULATORS

by

Chalongrath Pholsiri, B.Eng., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2004

Dedication

I dedicate this work to my parents, Somboon and Pongpen Pholsiri, and my sister, Piruna Pholsiri. I also dedicate this work to my wife, Supanee Maneerat, and my daughter, Tanyaton Layla Pholsiri.

Acknowledgements

I would like to express my deep gratitude to Dr. Delbert Tesar for his support, supervision, and advice throughout my research. I wish to thank Dr. Chetan Kapoor for creating OSCAR and RoboWorks and for all his advice and assistance. Many thanks go to all my UTRRG colleagues who have given me support and gone through the same experience throughout these years of my research endeavor.

I would like to thank my parents for always believing in me and for always giving me full moral support no matter the situation. Last but certainly not least, my special thanks go to my wife for her understanding and always being there for me and my daughter who brought to the world with her the joy and happiness I did not know existed.

CHALONGRATH PHOLSIRI

TASK-BASED DECISION MAKING AND CONTROL OF ROBOTIC MANIPULATORS

Publication No._____

Chalongrath Pholsiri, Ph.D.

The University of Texas at Austin, 2004

Supervisor: Delbert Tesar

Co-Supervisor: Chetan Kapoor

Redundancy affords an opportunity to deploy robotic manipulators in a broader variety of tasks owing to extra system resources. Decision making is a procedure that manages these extra resources and utilizes them in a meaningful fashion. Decision making is inherently complex and this complexity is a major hindrance to wider acceptance of redundant manipulators in various robotic applications. This report aims directly at decreasing this complexity by incorporating the concept of task requirements as an integral part of the decision making process.

Most traditional Redundancy Resolution Techniques (RRTs) place an undue burden on the user by forcing him to choose a set of performance criteria and assign relative importance to each criterion for a given task. The Task-Based Redundancy Resolution (TBRR) approach presented here changes all that by directly utilizing robotic task requirements in terms of speed, force, and accuracy in

the decision making process itself. TBRR searches the null space for configurations that comply with system constraints such as joint travel limits, singularities, and obstacles. TBRR then determines configurations that satisfy task requirements by estimating in real-time robot capabilities using a newly-developed technique called the Vector Expansion method. Finally, TBRR selects, among these configurations, the best solution according to efficiency or any other desired criterion. As a result, TBRR does not require a confusing chore of criteria fusion and thus is easier to use than traditional RRTs. Demonstrations on three geometrically different (6-, 7-, and 10-DOF) spatial robots show that TBRR provides 3% to 147% improvement over traditional RRTs as far as satisfying task requirements. A preliminary effort at integrating a learning method to help determine proper values of subjective parameters significantly reduces the human trial and error effort while showing a 3% improvement over the best set of hand-tuned parameters.

Another issue that has not been adequately addressed but is critical to redundant robotic manipulation is force control. Contact tasks, which represent a large portion of robotic tasks, cannot be effectively performed without one form of force control or another. This report illustrates the feasibility of integrating existing force control methods with the TBRR approach. The end result of this research is a task-based decision making and control framework that should enhance task performance of robotic manipulators operating with redundancy in a wider range of robotic tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Throughout the history of mankind, humans have tried to build machines that can mimic human versatility and abilities. Robots[1] have emerged from this very quest. Because humans interact with the world and perform a wide variety of tasks with their arms and hands, it is no accident that most robots possess mechanisms that resemble the human arm and hand. Humans, through experience, have learned to take advantage of their arms' versatility and dexterity. One can reach his hand to grab a fork while avoiding a hot pot on the stove. One can position his hand in such a way that maximizes the ability to exert force when pushing an incapacitated car. A surgeon can perform highly complex motions during surgery. Etc.

Many have seen the advantages that the human arm's redundancy can offer and have tried to build that into robotic systems. Redundancy, in this context, is defined as having more inputs than those required to produce the desired output. As such, traditionally non-redundant robots, e.g. most 6-DOF commercial robots, can be considered redundant too if their tasks at hand require fewer DOFs than the robots possess. Redundancy implies reconfigurability of the joint configurations without changing the position of the robot's end-effector, being able to create a so-called *self motion*. It allows performance enhancement during operation based on assessments of the assigned task, the environment, and the system states through selection of a set of options that optimizes or simply enhances a performance objective.

---

[1] The terms 'robots' and 'manipulators' are used interchangeably in this report and they both refer to serial manipulators.

With redundancy, however, comes complexity. The straightforward inverse kinematics problem becomes a more complicated optimization problem with the introduction of redundancy. Criteria that measure the best allocation of system resources in terms of the chosen performance objectives must be defined. For instance, criteria such as those for avoiding obstacles, maximizing mechanical advantage or velocity ratio, or minimizing the manipulator's kinetic energy have been defined. Early research efforts had focused on defining a particular criterion and devising an algorithm that would optimize the criterion in a computationally efficient manner. With several popular criteria defined and with recognition that in a typical task multiple criteria are usually needed, researchers later shifted the emphasis to multi-criteria optimization when solving the inverse kinematics problem.

As it will become apparent from the discussions in the literature survey section, the *Multi-criteria Redundancy Resolution Problem* (MRRP) has recently transformed into a problem of finding a proper *blend* of multiple criteria for a given instance. With literally tens of criteria to choose from (many of which are practically cryptic to non-expert users), the user is faced with an undue burden of choosing the right mix of these criteria. Many researchers have tried to alleviate this burden by coming up with *strategies* that dynamically adjust the weight of each criterion. While these strategies have shown promise, almost all of them fail to take into account the very basis of robotic tasks – *task requirements*.

The work presented here in this report offers a radically different approach to solving the MRRP. Instead of focusing on how and when to combine a set of criteria whose physical meanings are questionable at best, the approach we take here will ignore them altogether and consider task requirements in place of them. Only criteria with crisp and clear physical meanings will be included in a systematic fashion. We truly believe that this radical approach will take the full advantage of redundancy, give the user a full control without overwhelming

him/her, and facilitate rapid deployment of the system for variety of different tasks. In addition, the solution presented here should be generalized enough for any serial manipulator, thus enabling the use of modular, reconfigurable robots.

In most real-world robotic applications such as assembly or deburring, robots must interact with the environment. Whenever performing such tasks, manipulators are subject to interaction forces. The interaction forces must be controlled in such a manner that the manipulators comply with the environmental constraints. These constraints may be represented by flat or curved surfaces, deformable objects, work functions, etc. The existence of redundant robots complicates the management of the required level of the force control even further. Most force control schemes have focused on non-redundant robots until recently. Even then, redundancy has not been given its deserved attention. The redundancy resolution techniques that have been typically incorporated in force control are neither generalized nor expandable. On the other hand, the kinematic redundancy problem has been thoroughly studied in the past few decades. Some multi-criteria redundancy resolution techniques that are generalized and expandable have been proposed. However, all of these techniques do not adequately take into account the dynamics and control issues when it comes to solving the redundancy problem.

1.1 Background

This section presents a brief discussion of background materials to familiarize the reader with fundamentals of robotics and to prepare the reader for materials that will follow in the literature review section.

A robotic manipulator is essentially composed of rigid bodies – called *links* – that are connected to each other by *joints* in either an *open-chain (serial)*, *closed-chain (parallel)*, or *hybrid* fashion[2]. A joint usually consists of an actuator and its

[2] Only serial manipulators will be studied in this research.

axis of motion. The motion may be a revolution around the axis or a translation along the axis. These joints are known as *revolute* and *prismatic*, respectively. A manipulator is characterized by the number of actuated joints it possesses, called Degrees Of Freedom (DOF). The manipulator is often required to interact with its surroundings with its final link, called the *end-effector* (EEF). A point on the end-effector called *tool point* is the primary location of interaction between the manipulator and its environment.

The location of the tool point can usually be described as a nonlinear function of the joint variables.

$$x = f(\phi) \tag{1.1}$$

where $x \in \mathcal{R}^m$ denotes the EEF (position) variables, $\phi \in \mathcal{R}^n$ the joint variables, and $f : \mathcal{R}^n \to \mathcal{R}^m$ a set of nonlinear, transcendental, differentiable mappings. Eq. (1.1) is known as the forward kinematics equation.

Differentiating Eq. (1.1) once yields

$$\dot{x} = J(\phi)\dot{\phi} \tag{1.2}$$

where $J(\phi) = \partial f / \partial \phi \in \mathcal{R}^m \times \mathcal{R}^n$ is the Jacobian matrix. The nonlinear expression of Eq. (1.1) is thus converted into a linear equation in velocities.

Due to its importance in the operation of the robot, the *Inverse Kinematics Problem* (IKP) – which is defined as: given the desired EEF position, velocity and/or acceleration, determine the corresponding joint positions, velocities, and/or accelerations – is normally of more interest. Depending on the geometry of the robot, IKP may be resolved in either an analytical (closed-form solutions) or numerical fashion. Closed-form solutions (of some complexity) have the advantage of being more accurate, computationally more efficient, and less prone to numerical instability. However, only geometrically simple robots of up to 6 DOFs possess closed-form solutions. Numerical solutions, on the other hand, can be applied to any robot. The University of Texas Robotics Research Group

(UTRRG) has developed a numerical inverse kinematics solution for general serial manipulators [Hooper and Tesar, 1994].

For redundant manipulators ($n>m$), an infinite number of possible inverse solutions exist for a given EEF position. This phenomenon is illustrated by *self motion* in which the joints can move while the EEF remains stationary (Figure 1-1). Mathematically speaking, there exists a null space, which is a set of possible joint configurations that realize the same EEF position, in the joint space. One can search the null space to find a potential solution that enhances the robot performance while satisfying the EEF position constraints. IKP for redundant robots is named the *Redundancy Resolution Problem* (RRP) and can be solved by a *Redundancy Resolution Technique* (RRT). Many of the existing RRTs will be thoroughly investigated in Chapter 3.

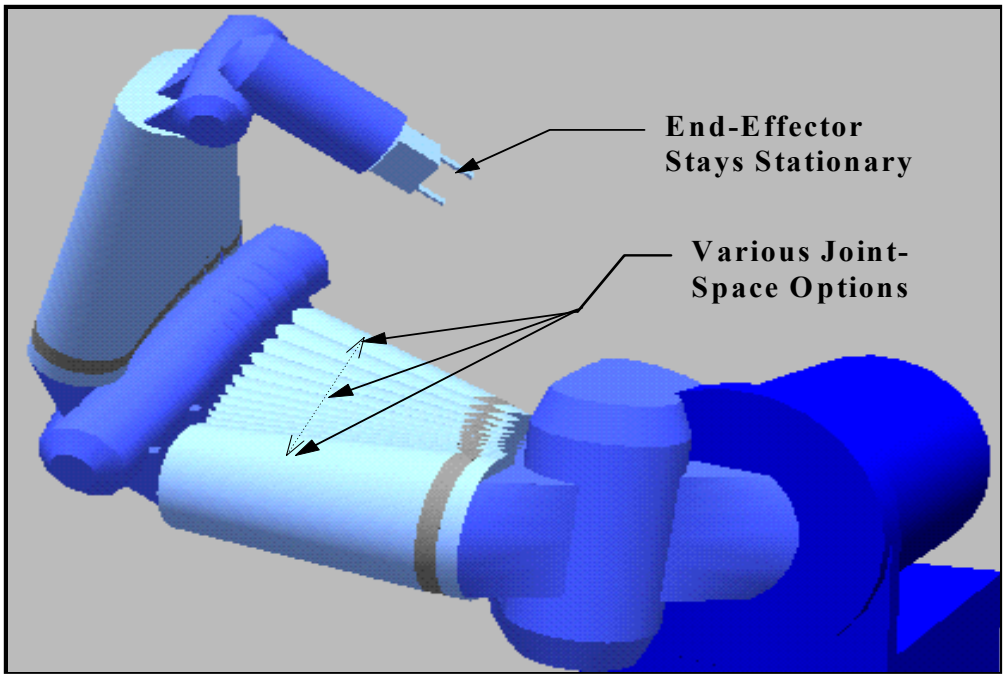


Figure 1-1: A redundant robot demonstrating "self-motion" (adapted from Hooper and Tesar [1994]).

1.2 Literature Survey

Decision making is possible in redundant mechanical systems due to the fact that they have extra resources that can be allocated, enabling choices to be made. The decision maker needs to compare all the options and then decide on the best one. In RRP, this is equivalent to constructing an objective function, evaluating the function, and then picking the option that optimizes (minimizes/maximizes) the function. If the objective function consists of only one criterion, such as minimizing joint velocities or maximizing measure of transmissibility, then this process is straightforward.

One of the early adopted RRTs is the use of generalized inverses of the Jacobian. Two particular generalized inverses, namely pseudoinverse and inertia-weighted pseudoinverse, can be used to "minimize the two-norm of the joint velocity vector" and "minimize the joint kinetic energy," respectively [Whitney, 1969]. They provide a particular solution to the under-constrained set of equations. One cannot include any other performance criterion into the solution.

Liegeois [1977] extended the idea of generalized inverses by exploiting the null space possessed by the non-square Jacobian. The null space adds the homogeneous solution that does not contribute to the end-effector motion. By projecting the gradient of a performance function onto the null space, the obtained solution can decrease (or increase) and eventually locally minimize (or maximize) the performance function value. This method was later known as the *Gradient Projection Method* (GPM).

The configuration control method [Seraji, 1989], which is a superset of the extended Jacobian method [Baillieul *et al.*, 1984], avoids the use of generalized inverses by adding *n-m* additional kinematic functions to the forward kinematics relations, thus squaring the Jacobian. The augmented Jacobian can then be inverted. The addition of constraints, however, adds algorithmic singularities, in addition to kinematic singularities, and can restrict the inversion. Moreover, since

extending the Jacobian is limited to *n-m* constraints, it is not expandable to consider an arbitrary number of performance criteria.

Hooper and Tesar [1994] developed a Direct Search (DS) method for first solving the inverse kinematics problem of non-redundant robots. The method uses the forward position solution to guide the search for the inverse solution. The search begins with an estimated solution, which serves as a base configuration. Then small joint perturbations are introduced to generate new configurations and the "error" is calculated at each of these new configurations. A new base configuration that reduces the error is then chosen. This process continues until the search finds a solution, i.e. the error is zero. The search is direct in the sense that it does not use any derivatives and hence it is robust, especially at or around mathematical singularities. This method was then extended to solving redundancy resolution with multiple criteria. This problem is solved by using any six joints to satisfy the end-effector position requirements. The remaining joints are used to generate joint-space options. Criteria are then evaluated at each of these options and the options are ranked based on some metrics. The option with the best ranking is then selected.

Since the completely generalized direct search method is very computationally expensive, a closed-form inverse kinematics solution is used in the 6-DOF substructure to meet the end-effector constraints. This however reduces the generality of the method. Kapoor and Tesar [1996] and Kapoor *et al*. [1998] later developed the *Performance-Based Hybrid Generalized Inverse* by utilizing a generalized inverse for the 6-DOF substructure instead of the closed-form inverse. This enables the DS method to perform in real-time applications and yet maintain an expanded generality.

Solving the inverse kinematics problem at the velocity level, Cheng *et al.* [1992] formulated the RRP as a quadratic programming (QP) problem. QP is a problem where one attempts to optimize a quadratic objective function subject to

linear equality/inequality constraints. By exploiting some properties of the Jacobian, they were able to develop an efficient algorithm to solve this QP problem.

When faced with optimizing multiple criteria, many researchers have resorted to the weighted sum method as a way to combine those criteria into a single objective function. Due to the popularity of the weighted sum method, much attention has been paid to devising the *best* weighting scheme. Weighting strategies could use either fixed weights or time-varying weights. Time-varying (or dynamic) weighting means that each weight varies over time during an operation. The rationale for using time-varying weighting is that because a robotic system is dynamic in nature, at each instant criteria have different degrees of importance to the overall objective.

McGhee *et al.* [1994] proposed a probability-based weighting technique. In this method, probability density functions are constructed and used as weights. However, the construction process of these probability density functions is strenuous as it involves collecting data for each criterion at every point along a set of trajectories that systematically span the entire workspace. In their experiment, 750,000 sample points were used to gather the data needed to construct such functions.

Hanson and Tolson [1995] used a fuzzy logic supervisor to adjust the relative weights when optimizing for multiple criteria. One advantage of using fuzzy logic is that linguistic rules are intuitive and easy to understand. For each relative weight, a membership function is required. In their membership function construction, three parameters are needed to determine only one weight. Even though the authors argued, "... choosing the three membership function parameters is easier than the single fixed weight," the fact remains that the assignment of these parameters by the operator is still ill-defined and trial and error is still needed.

Cheng *et al.* [1996 and 1997] devised a decision making strategy for weight adjustments in order to cope with multiple criteria. In this strategy, higher-priority criteria or "masters" can modulate the weights of lower-priority criteria or "slaves." To implement this strategy, they developed a *parallel scheme* and later an *improved parallel scheme*. For a certain criterion, an exponential function of a *normalized reciprocal necessity index*, defined as the ratio between the criterion value and its minimum, was used to determine the weight of the criterion. With this scheme, the weights could be dynamically adjusted according to the criteria's levels of priority and their values at run-time. Although their decision making strategy stated that "When higher-priority goals are not necessary to be accomplished, the weights of these higher-priority goals should be released (reduced) such that lower-priority goals can be achieved more easily", their implementation of the parallel scheme was not carried out in this way. In this scheme, the weights of masters would *always* be higher than those of slaves no matter what. This means that the masters were always considered more important even when they were not necessary.

Despite the popularity of the weighted-sum method, it has some serious drawbacks that are noteworthy. First, criteria are incommensurate and it is not technically correct to add numbers of different units. Secondly, criteria vary significantly in magnitudes. Some criteria range between 0 and 500,000 and some between 0 and 1. Without normalization, high-average criteria can easily dominate the solution. Third and perhaps most important, optimizing the weighted sum does not necessarily mean that all or most of the criteria are acceptable. For example, let's assume that we want to optimize two criteria: Joint Range Availability (JRA) and Measure of Transmissibility (MOT). Let's also assume that these two criteria have been normalized such that the value of zero indicates the worst case and one the best. In case of JRA, zero means joints are close to or at the joint limits and one means joints are in the middle of the travel ranges. For MOT, zero means the robot

is at a singularity and one means the robot is very far away from singularities[3]. Normally, we would like to maximize both criteria. Let's consider two hypothetical cases. In case A, we have JRA of 0.01 and MOT of 0.95 and case B has JRA of 0.4 and MOT of 0.4. We assign the value of 0.5 to the weights of both criteria because we think that both are equally important. According to the weighted-sum method, case A is better since the composite index is 0.48 compared to 0.4 for case B. In reality though, case B would be a better choice since it places the robot at a safer configuration in regards to joints exceeding their limits. Even though its MOT is not as good as in case A, the robot is still not near a singularity.

Contending that weight adjustment was not the way to resolving MRRP, Cetin and Tesar [1999] sought the help of *Multi-Criteria Decision Making* (MCDM) theory or multiple objective nonlinear programming, particularly goal programming. They developed a *Compromise Solution* (CS) method and introduced a concept of *Maximum Allowable Variation (MAV)*. The user specifies MAV for each criterion and CS then determines the solution that minimizes the norm of the weighted sum of the percent deviations of all criteria while satisfying the trajectory constraints and attempting to keep each deviation under its MAV. One major benefit of this method is that it takes the normalization issue of the criteria out of the equation. A criterion with a lower MAV is considered more important than one with a higher MAV. This method, however, simply changes the user's responsibility from assigning weights to assigning MAVs. Although this method also allows the user to interactively change the MAV values online to deal with conflicts that may occur, assigning appropriate values for MAVs is still no more trivial than assigning weights.

In dealing with multiple criteria, Nakamura *et al*. [1987] and Nagamura [1991] presented *task prioritization*, where a task is divided into subtasks of

---

[3] Sometimes, the task demands that we approach certain singularities.

different orders of priority. Subtasks with higher priority are to be performed first and less important subtasks are satisfied only in the null space of the higher-priority tasks. Siciliano and Slotine [1991] proposed a recursive formulation of the task prioritization technique. As many subtasks can be added as seen fit by the operator. However, when the lower-priority tasks cannot be fulfilled without preventing the high-priority tasks from being realized, an *algorithmic singularity* occurs. It is also difficult to change the orders of priority in real time, rendering this method less adaptable.

Pryor and Tesar [1999] examined the effectiveness of incorporating multiple criteria and concluded that an excessive number of criteria included in the redundancy resolution scheme could aggravate the efficacy of the scheme. Pryor and Tesar [2002] later developed a concept of *critical boundaries*, a strategy of including criteria only when they are critical to the completion of the task at hand. Within this concept, performance criteria are divided into two major groups: constraint-based and non constraint-based criteria. Constraint-based criteria are ones that assist the manipulator in avoiding physical or mathematical constraints, such as joint travel limits, joint velocity limits, singularities, or obstacles. Therefore, a critical boundary is defined for each such criterion. Only when the criterion value crosses the critical boundary is the criterion included in the overall objective function. If no constraint-based criteria are critical at the time, then efficient operation can become the sole objective for the RRT to evaluate. With this approach, they ran an exhaustive simulation with approximately 38,000 set points and no more than 3 criteria were active at any given time. This simple approach enables the RRT to function more efficiently by limiting the number of criteria into the RRT at any time. With strategically defined critical boundaries for several crucial criteria, the user is left with a fairly intuitive task of choosing how aggressive or conservative he wants to regard each criterion. This approach can be thought of as a special case of dynamic weight adjustments, where the weight for

each criterion is binary (either 0 or 1). When a criterion is critical, its weight is set to 1. Otherwise, it is set to 0. When no criterion is critical, the efficiency criterion's weight is set to 1. In essence, this approach does not address the issue of criteria fusion.

Tisius *et al.* [2004] took a new empirical approach in attempting to get a better understanding of performance criteria and multi-criteria redundancy resolution. The manipulator's performance maps are generated empirically via computer simulation by running the robot over the course of a prescribed path. These performance maps can then be used to normalize performance criteria such that they are comparable in magnitudes. This normalization process "allows the user to make valid comparisons, and aids the RRT's ability to prioritize" [Tisius *et al.*, 2004]. This empirical approach also allows RRTs to employ *global normalization* instead of the current use of *local normalization*. Global normalization was shown to be more stable, reflect the physical state of the robot more accurately, and handle conflicts among multiple criteria more effectively.

Most of the criteria that have been developed can be described as to assist the robot in performing a secondary task such as avoiding obstacles or singularities. The question regarding control performance has never been addressed. Choi [1999] proposed a joint disturbance torque criterion that can be used in the independent joint controlled robot. The joint disturbance torque includes the Coriolis and centripetal effects, gravity torque, and the effect of non-diagonal elements of the inertia matrix. Minimizing this torque can reduce the joint tracking error in the independent joint control system. This criterion is, however, very complex and a function of joint positions, velocities, and accelerations. The paper only demonstrated its use on a 3-jointed planar robot and failed to mention how to extend this method to spatial robots.

The notion of task oriented optimization for redundant manipulators is presented in [Chiu, 1988] based on the task ellipsoid concept and robot's velocity

and force transmission ratios. The shape of a task ellipsoid is determined from the relative force and motion requirements along the task directions. The idea is to "minimize the task characteristics between the actual [manipulator's] force ellipsoid and the task ellipsoid."

1.3 Research Objectives

The objective of this research is to address the aforementioned problems encountered in the world of redundant robots. There exists a need for an approach that bridges the gaps among mathematical formulations (robot models & performance criteria), control schemes, and task requirements when redundant robots are involved. It is the purpose of this research to fill those gaps in a systematic manner.

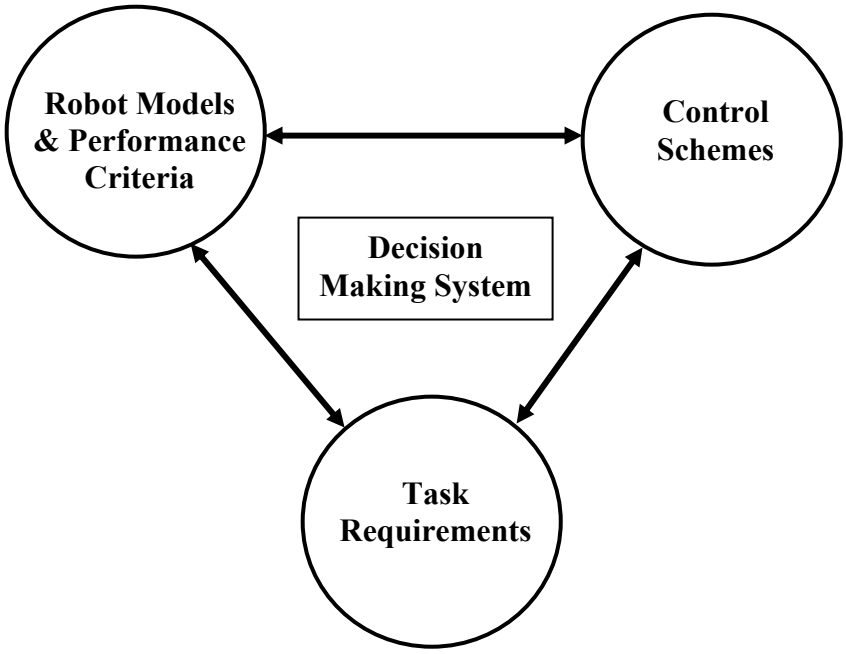


Figure 1-2: Task-Based Decision Making Framework.

UTRRG has had success developing performance criteria and generalized criteria-based inverse kinematics algorithms for serial robotic manipulators. The multi-criteria inverse kinematics methods developed by UTRRG are completely

generalized, expandable, and efficient. It is the intention of this research to construct a *task-based controller* for redundant manipulators based upon the existing multi-criteria inverse kinematics. The task-based controller is a controller that unifies position and force control of redundant manipulators and takes task requirements as the central component when resolving redundancy. In order to construct such a criteria management approach, the following tasks will be pursued:

- Improve the performance of the existing multi-criteria inverse kinematics method currently in use at UTRRG.
- Describe tasks as compilations of numerical task requirements in terms of speed, force, and accuracy.
- Develop a redundancy resolution strategy that incorporates task requirements as an integral part. This method will be called Task-Based Redundancy Resolution (TBRR) algorithm.
- Investigate existing force control schemes and determine the one that is suitable for integrating with TBRR to construct the task-based decision making approach.

Finally, all redundancy resolution techniques and control laws have system parameters whose values must be properly chosen for the system to perform well. The task-based approach is no exception. It is often not straightforward to select appropriate values for these system parameters. In order to make the task-based approach more accessible to non-expert users, we must facilitate the selection process of the values of the system parameters.

1.4 OSCAR Summary

OSCAR (Operational Software Components for Advanced Robotics) is an object-oriented framework of the operational software layer for controlling intelligent machines developed and used at UTRRG [Kapoor and Tesar, 1996].

Figure 1-3 illustrates how OSCAR fits in the overall scheme of robotics software. OSCAR libraries contain essential components needed for developing control software for serial manipulators including forward and inverse kinematics, dynamics, performance criteria, and decision making. Major attributes of OSCAR are generality, extensibility, and reusability.

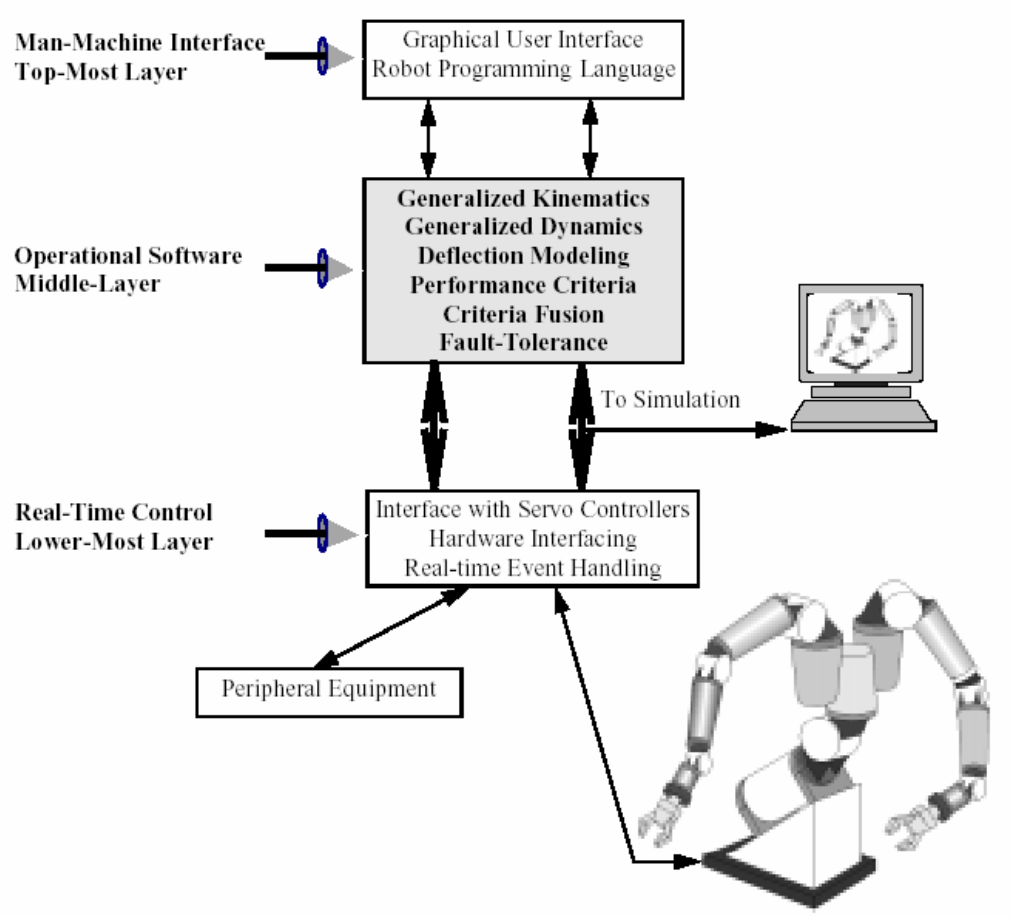


Figure 1-3: Three layers of robotics software. OSCAR represents the middle layer [Kapoor and Tesar, 1996].

OSCAR will be used throughout this research as both a development tool and a test environment for simulated robots. As a development tool, OSCAR provides to all the basic computations necessary for developing more advanced control and decision-making components. This research will undoubtedly

contribute to functionality enhancement and, perhaps, additional libraries to OSCAR. As a test environment, OSCAR provides computation for kinematics and dynamics of generalized serial manipulators. Therefore, it is trivial to simulate kinematic and/or dynamic behaviors of any serial manipulator using OSCAR.

1.5 Chapter Summary and Report Overview

This introductory chapter has familiarized the reader with some basic fundamentals of robotics, described the problem of decision making for the control of redundant robots, and briefly discussed what has been done in the research community to cope with the problem. The significant issues associated with meeting task requirements and force control for redundant manipulators were identified and their incorporation in the decision making process was posed as the purpose of this research.

Manipulator modeling and controls are essential for the development of the problem solutions that will follow. Therefore, their detailed discussions will be presented in Chapter 2. Chapter 3 investigates several existing RRTs that have been proposed in the literature. The emphasis will be on the ones developed at UTRRG. An improvement on one particular RRT that is being used at UTRRG will also be introduced in Chapter 3. In addition, this chapter reviews numerous performance criteria that have been recognized as being critical to operations of robotic systems. These criteria will be discussed in detail, including their potential uses in RRTs when completing a given task.

Chapter 4 introduces the concept of task requirements in generic robotic tasks and develops an accurate estimate of the robot capability associated with each task requirement using a new technique called the Vector Expansion (VE) method. The robot capability analysis can benefit a number of different robotic applications from design to real-time operation.

In Chapter 5, a new strategy of resolving redundancy called Task-Based Redundancy Resolution (TBRR) is proposed. TBRR is unique in the sense that it revolves around exploiting redundancy to help the robot perform the task at hand successfully by utilizing the task requirement specifications and robot capability estimates obtained via the VE method. A learning algorithm of some TBRR system parameters is also introduced. This learning process helps determine good values of system parameters, thereby eliminating the usual approach of trial and error. The effectiveness of TBRR and parameter learning is presented by means of computer simulations of several manipulators of different geometries. Simulations are diverse in terms of requirements and complexity to highlight the performance and versatility of TBRR.

Chapter 7 deals with integrating force control into the TBRR framework. Several existing force control methods are first presented. Pros and cons of each method will be considered with the intention of determining an appropriate force control strategy that can seamlessly integrate with TBRR. Then, the integration process is detailed and issues discussed. Lastly, Chapter 8 concludes this research by summarizing the key contributions that this research provides and then outlines future research areas that needed to be conducted for this research and robotics in general to be of utmost practicality.

CHAPTER 2

MANIPULATOR MODELING AND CONTROL

A mathematical model of a manipulator is essential in describing its motions and dynamic responses when excited. This chapter covers kinematic, dynamic, and compliance models and control methods for serial manipulators and it will serve as a foundation and a reference for the remainder of this work.

2.1 Generalized Kinematic Influence Coefficient Models

This section reviews the manipulator modeling methods first developed at University of Florida and later at University of Texas Robotics Research Group (UTRRG). The detailed formulations of these models can be found in Thomas and Tesar [1982], Freeman and Tesar [1988], and Hernandez and Tesar [1996]. These models can be used to describe kinematics, dynamics, and compliance of any serial manipulator. They are based on the concept of *Kinematic Influence Coefficients (KICs)*.

2.1.1 Kinematics

In the study of kinematics of a serial manipulator, we attempt to determine the relationships between the motion of a given point (most notably the EEF) on the manipulator and the motion of the manipulator joints, assuming that the manipulator links are rigid.

2.1.1.1 Geometric Representations of Rigid Manipulator Links

A joint-link pair in a manipulator can be described by at least 4 parameters. The most common notation of these 4 parameters is called the *D-H parameters* [Denavit and Hartenberg, 1955]. Depending on the convention of numbering the

coordinates, two representations of the D-H parameters are widely used. This report follows the representation by Craig [1989], which is also used in OSCAR[4] and is illustrated in Figure 2-1.

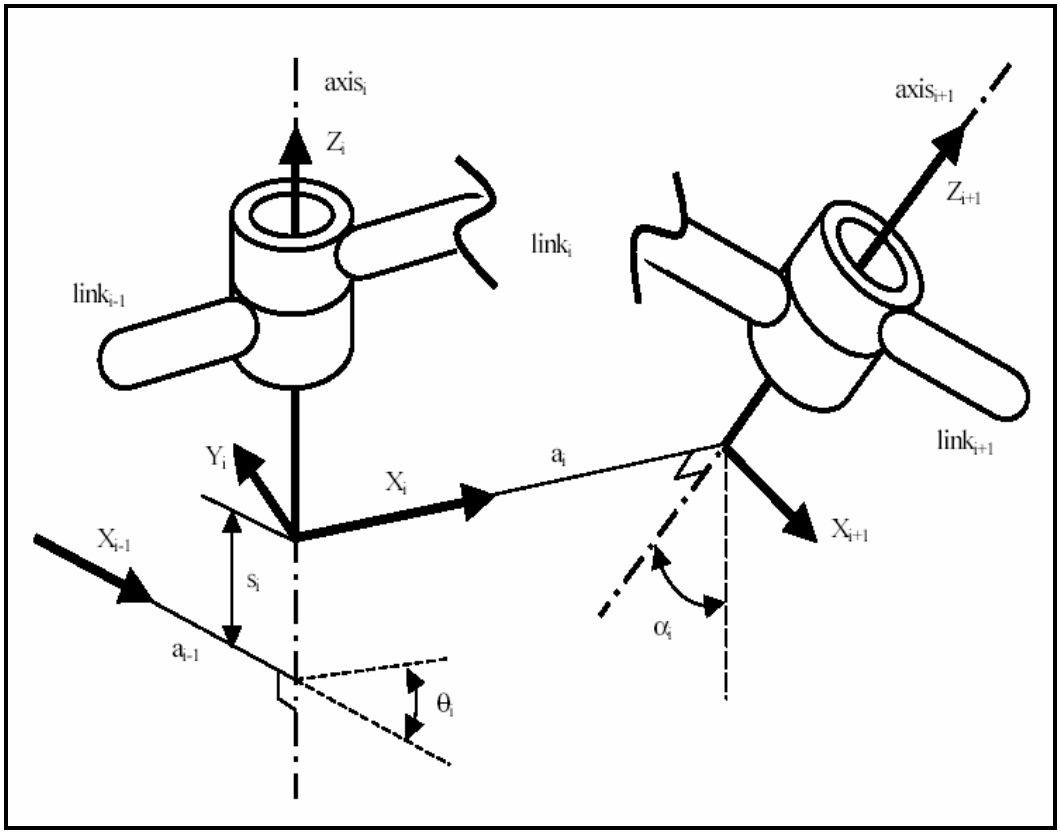


Figure 2-1: D-H Parameters

The link parameters in Figure 2-1 are designated as follows:

- a_i (link length): the distance from Z_i to Z_{i+1} measured along X_i;
- s_i (joint offset length): the distance from X_{i-1} to X_i measured along Z_i;
- θ_i (joint angle): the angle between X_{i-1} and X_i measured about Z_i; and

---

[4] Craig's representation was the only D-H parameters notation supported in OSCAR 1.0. However, OSCAR has recently been updated to version 2.0. In OSCAR 2.0, the other representation of D-H parameters is also supported.

- α_i (twist angle): the angle between Z_i and Z_{i+1} measured about X_i.

Thomas and Tesar [1982] presented a notation that closely parallels D-H notation but requires more parameters. This may seem redundant but it "increases convenience when determining position coordinates of *any* point in the link." [Thomas and Tesar, 1982] Thomas and Tesar's notation is illustrated in Figure 2-2.

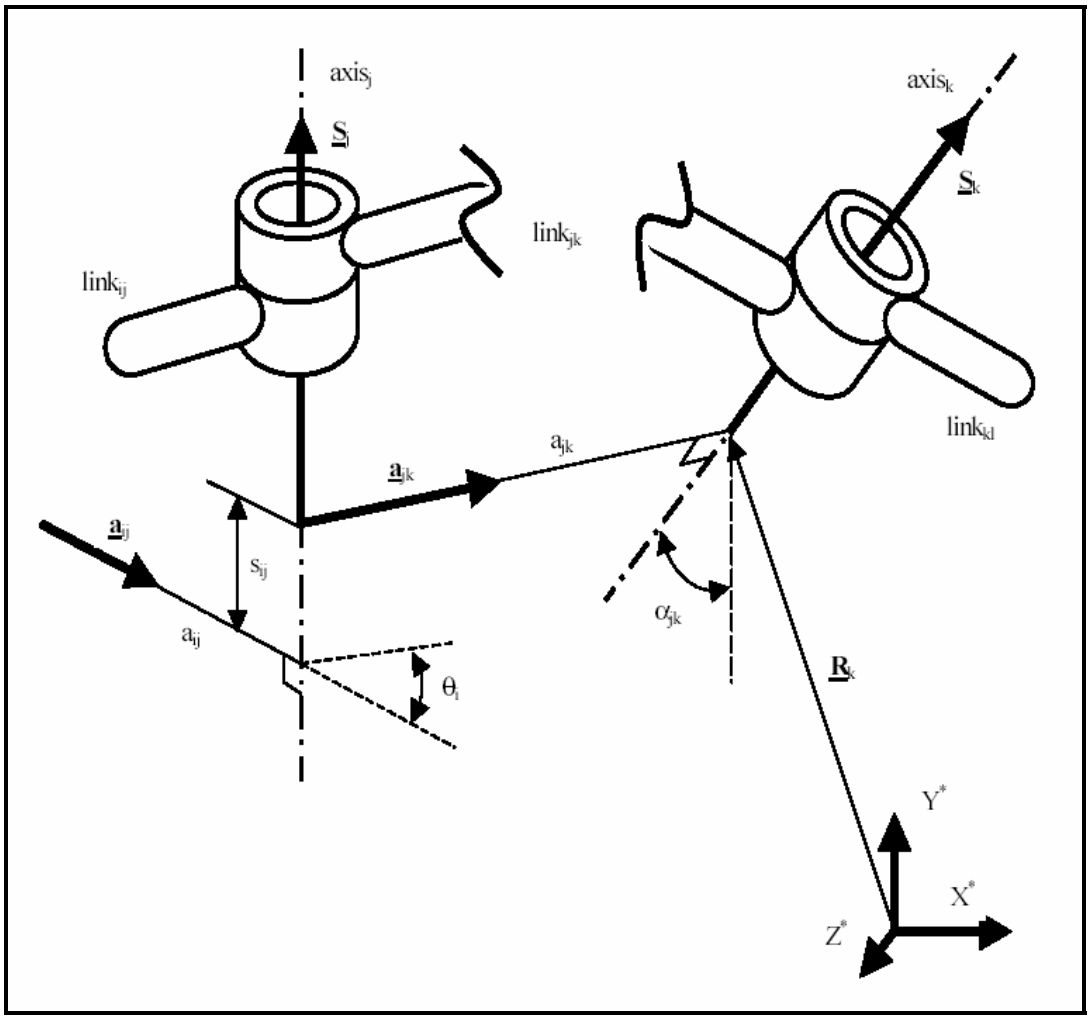


Figure 2-2: Thomas and Tesar Notation

The symbols in Figure 2-2 are designated as follows:

- $\left[\mathbf{X}^* \quad \mathbf{Y}^* \quad \mathbf{Z}^*\right]$ is an orthogonal basis for the fixed base frame that is aligned with the first link such that $\mathbf{Z}^* = \hat{\mathbf{S}}_1$;
- $\hat{\mathbf{S}}_j$ is the unit vector along joint axis *j*;
- S_j is the offset distance along a joint axis *j* between connected links;
- $\hat{\mathbf{a}}_{jk}$ is the unit vector along the common perpendicular vector of axes *j* and *k*;
- a_{jk} is the perpendicular distance along $\hat{\mathbf{a}}_{jk}$ between axes *j* and *k*;
- θ_j is the angle of joint *j*;
- α_{jk} (twist angle) is the angle between axes *j* and *k*;
- $\mathbf{R}_j$ is the origin of the moving system relative to the world frame. The point of interest in the moving system is located at the intersection of $\hat{\mathbf{S}}_j$ and $\hat{\mathbf{a}}_{jk}$;
- $\mathbf{P}^j$ (not shown) is a point located anywhere in the j^{th} moving system defined in a local frame $\left[\mathbf{X}^j \quad \mathbf{Y}^j \quad \mathbf{Z}^j\right]$. **P** is the same point located in the base frame.

If joint *j* is revolute, then θ_j is the input ϕ_j and if the joint is prismatic, then ϕ_j is S_j.

2.1.1.2 First-Order Influence Coefficients

Let $\mathbf{u} \in \mathcal{R}^m$ represent the output position vector (e.g. it could be the displacement of a point of interest) and $\phi \in \mathcal{R}^n$ the vector of generalized coordinates. Then, **u** can be described as a function of ϕ.

$$\mathbf{u} = \mathbf{f}(\phi) \tag{2.1}$$

By taking the time derivative of Eq. (2.1), we can obtain the output velocity vector as

$$\begin{aligned}\dot{\mathbf{u}} &= \frac{\partial \mathbf{u}}{\partial \phi}\frac{d\phi}{dt} \\ &= \left[G_\phi^u\right]\dot{\phi}\end{aligned} \tag{2.2}$$

The matrix $\left[G_\phi^u\right] \in \mathcal{R}^{m\times n}$, called the Jacobian, is a geometrically dependent structure relating the joint velocities to the output velocities in a linear fashion. In general, the Jacobian for a point P on a link jk consists of two sub-matrices containing the translational and rotational components.

$$\left[G_\phi^u\right] = \begin{bmatrix}\left[G_\phi^P\right] \\ \left[G_\phi^{jk}\right]\end{bmatrix} = \begin{bmatrix}\mathbf{g}_1^P & \mathbf{g}_2^P & \cdots & \mathbf{g}_n^P \\ \mathbf{g}_1^{jk} & \mathbf{g}_2^{jk} & \cdots & \mathbf{g}_n^{jk}\end{bmatrix} \tag{2.3}$$

where

$$\mathbf{g}_i^P = \begin{cases}\mathbf{S}_i \times \mathbf{R}_i & \text{if } i \le j \text{ and joint } i \text{ is revolute} \\ \mathbf{S}_i & \text{if } i \le j \text{ and joint } i \text{ is prismatic} \\ 0 & \text{if } i > j\end{cases} \tag{2.4}$$

$$\mathbf{g}_i^{jk} = \begin{cases}\mathbf{S}_i & \text{if } i \le j \text{ and joint } i \text{ is revolute} \\ 0 & \text{otherwise}\end{cases}. \tag{2.5}$$

It should be noted here that the Jacobian at the end-effector $\left[G_\phi^e\right]$ is usually referred to as J in the literature. Due to its simplicity, this notation will also be adopted here in this report.

2.1.1.3 Second-Order Influence Coefficients

Differentiating Eq. (2.2) with respect to time, we obtain the output acceleration vector

$$\begin{aligned}\ddot{\mathbf{u}} &= \frac{\partial \mathbf{u}}{\partial \phi}\ddot{\phi} + \dot{\phi}^T \frac{\partial^2 \mathbf{u}}{\partial \phi^2}\dot{\phi} \\ &= \left[G_\phi^u\right]\ddot{\phi} + \dot{\phi}^T\left[H_{\phi\phi}^u\right]\dot{\phi}\end{aligned}. \tag{2.6}$$

The Jacobian relates joint accelerations to the EEF accelerations. The tensor $\left[H^{u}_{\phi\phi}\right] \in \mathcal{R}^{m\times n\times n}$, known as the Hessian array, relates Coriolis and centripetal effects to the EEF accelerations. Similarly to the Jacobian, the Hessian can also be divided into translational and rotational component tensors, where each plane is represented as

$$\left[H^{u_i}_{\phi\phi}\right] = \begin{bmatrix} \left[H^{e_i}_{\phi\phi}\right] \\ \left[H^{jk_i}_{\phi\phi}\right] \end{bmatrix}. \tag{2.7}$$

Each translational component is calculated from

$$\left[H^{e_i}_{\phi\phi}\right]_{:,r,c} = \begin{cases} \mathbf{S}_r \times \left(\mathbf{S}_c \times \left(\mathbf{P}_i \times \mathbf{R}_c\right)\right) & \text{if } r \le c \le i \text{ and joints } r \text{ and } c \text{ are revolute;} \\ \mathbf{S}_c \times \left(\mathbf{S}_r \times \left(\mathbf{P}_i \times \mathbf{R}_r\right)\right) & \text{if } c \le r \le i \text{ and joints } r \text{ and } c \text{ are revolute;} \\ \mathbf{S}_r \times \mathbf{S}_c & \text{if } r \le c \le i \text{ and } r \text{ is revolute, } c \text{ is prismatic;} \\ \mathbf{S}_c \times \mathbf{S}_r & \text{if } c \le r \le i \text{ and } c \text{ is revolute, } r \text{ is prismatic;} \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{2.8}$$

For rotational components, each can be derived from

$$\left[H^{jk_i}_{\phi\phi}\right]_{:,r,c} = \begin{cases} \mathbf{S}_r \times \mathbf{S}_c & \text{if } r \le c \le i \text{ and joints } r \text{ and } c \text{ are revolute;} \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{2.9}$$

Here, i refers to a plane associated with a given output direction. The indices r and c refer to the row and column of a Hessian submatrix and each index ranges from 0 to n.

2.1.2 Dynamics

The joint torques due to the manipulator's inertia are given by

$$\tau^{I} = \left[I^{*}_{\phi\phi}\right]\ddot{\phi} + \dot{\phi}^{T}\left[P^{*}_{\phi\phi\phi}\right]\dot{\phi} \tag{2.10}$$

where $\left[I^{*}_{\phi\phi}\right]$ is the *effective inertia matrix* and $\left[P^{*}_{\phi\phi\phi}\right]$ the *inertial power array*. The first term includes the effects of the masses and moments of inertia while the

second term accounts for the centripetal and Coriolis effects. The effective inertia matrix and the inertial power array are defined respectively as

$$\left[I_{\phi\phi}^{*} \right] = \sum_{j=1}^{N} \left\{ M_{jk} \left[{}^{j}G_{\phi}^{c} \right]^{T} \left[{}^{j}G_{\phi}^{c} \right] + \left[G_{\phi}^{jk} \right]^{T} \left[\Pi^{jk} \right] \left[G_{\phi}^{jk} \right] \right\} \tag{2.11}$$

and

$$\begin{aligned} \left[P_{\phi\phi\phi}^{*} \right] = \sum_{j=1}^{N} \Big\{ & M_{jk} \left(\left[{}^{j}G_{\phi}^{c} \right] \circ \left[{}^{j}H_{\phi\phi}^{c} \right] \right) \\ & + \left(\left[G_{\phi}^{jk} \right]^{T} \left[\Pi^{jk} \right] \right) \circ \left[H_{\phi\phi}^{jk} \right] \\ & + \left[G_{\phi}^{jk} \right]^{T} \left(\left[G_{\phi}^{jk} \right]^{T} \circ \left[P^{jk} \right] \right) \left[G_{\phi}^{jk} \right] \Big\} \end{aligned} \tag{2.12}$$

Here, M_{jk} is the mass of link *jk*. $\left[{}^{j}G_{\phi}^{c} \right]$ is the translational Jacobian determined for the centroid of link *jk* and $\left[G_{\phi}^{jk} \right]$ the rotational Jacobian of link *jk*. $\left[\Pi^{jk} \right]$ and $\left[P_{\phi\phi\phi}^{*} \right]$ are the globally referenced inertia matrix and the globally referenced inertia tensor given by

$$\left[\Pi^{jk} \right] = \left[\mathbf{I}^{jx} \vdots \mathbf{I}^{jy} \vdots \mathbf{I}^{jz} \right] \tag{2.13}$$

$$\left[P^{jk} \right] = \begin{bmatrix} 0 & \mathbf{I}^{jz} & -\mathbf{I}^{jy} \\ -\mathbf{I}^{jz} & 0 & \mathbf{I}^{jx} \\ \mathbf{I}^{jy} & -\mathbf{I}^{jx} & 0 \end{bmatrix}. \tag{2.14}$$

The symbol '$\circ$' in Eq. (2.12) is the *generalized dot product* (see Freeman and Tesar [1988] for detailed discussion).

By applying the concept of virtual work, external loads, including gravity, can be mapped to joint torques in terms of the first-order influence coefficients as follows. Consider an external force $\mathbf{f}_{P}$ applied at point *P* and a moment $\mathbf{m}_{jk}$ applied on link *jk*. The total effective torque due to applied loads is

$$\tau^{L} = \sum_{P=1}^{\bar{P}} \left[G_{\phi}^{P} \right]^{T} \mathbf{f}_{P} + \sum_{j=1}^{n} \left[G_{\phi}^{jk} \right]^{T} \mathbf{m}_{jk} \tag{2.15}$$

where $\overline{P}$ is the total number of points of interest in the manipulator.

In most cases, the external loads only include the gravity and a load by the environment exerted at the end-effector. Let **L** denote a vector that contains both force and moment. The total torques acting on the actuators are therefore defined as

$$\tau_{\phi}^{total} = \left[I_{\phi\phi}^{*} \right] \ddot{\phi} + \dot{\phi}^{T} \left[P_{\phi\phi\phi}^{*} \right] \dot{\phi} + \sum_{i=1}^{n} \left[{}^{i}G_{\phi}^{cg} \right]^{T} \mathbf{L}_{g,i} + \left[G_{\phi}^{e} \right]^{T} \mathbf{L}_{e} . \tag{2.16}$$

The first and second terms represents the torques due to mass and inertia. The third term adds the gravitational effects and the fourth term contains the effect of external force at the end-effector.

2.1.3 Compliance Model

A compliance model of a manipulator here is a summary of the study by Hernandez and Tesar [1996]. The formulations presented here are based on the assumption that the manipulator is in a quasi-static state. Systems may be considered quasi-static if motions and loading produce negligible vibration. The compliance model is used in this report for the purpose of calculating and maximizing the stiffness of the system. There are two main sources of compliance in serial manipulators: joints and links. However, it has been shown that, for most industrial robots, the effects of joint compliance are significantly larger than those of link compliance [Berger and ElMaraghy, 1992]. The compliance formulations hereafter will therefore only consider the joint compliance.

The local joint compliance represents the total linear compliance of the joint transmission elements. The joint compliance parameters can be assembled into a diagonal matrix.

$$\left[C_{\phi} \right] = diag(c_1, c_2, \cdots, c_n) \tag{2.17}$$

where c_i is the individual compliance value (the joint deflection per unit torque) of joint *i*. The joint deflections due to a joint torque vector τ is

$$\Delta\phi^{\tau} = \left[C_{\phi}\right]\tau \tag{2.18}$$

If only an EEF load $\mathbf{L}_e$ is considered, applying the relationship $\tau = \left[G_{\phi}^{e}\right]^{T}\mathbf{L}_e$ yields the joint deflections due to the EEF load as

$$\Delta\phi^{L_e} = \left[C_{\phi}\right]\left[G_{\phi}^{e}\right]^{T}\mathbf{L}_e \tag{2.19}$$

Using the Jacobian to map the joint deflections to the EEF deflection results in

$$\Delta^{L_e} = \left[G_{\phi}^{e}\right]\left[C_{\phi}\right]\left[G_{\phi}^{e}\right]^{T}\mathbf{L}_e \tag{2.20}$$

The EEF stiffness matrix can then be defined as

$$\left[K_{EEF}\right] = \left(\left[G_{\phi}^{e}\right]\left[C_{\phi}\right]\left[G_{\phi}^{e}\right]^{T}\right)^{-1} \tag{2.21}$$

Because it is not uncommon that the weights of the robot's links represent the majority of the loads acting on the robot [Hernandez and Tesar, 1996], special care should be given to the deflection due to the manipulator gravity load. Let $\mathbf{F}_g$ denote a vector of the gravity forces acting on a given link and $\left[G_{\phi}^{cg}\right]^{T}$ be the first-order influence coefficients that relate the weights of the links to the torques at the actuators, the EEF deflection and joint deflections due to the weight are given respectively by

$$\Delta^{g} = -\left[G_{\phi}^{e}\right]\left[C_{\phi}\right]\left[G_{\phi}^{cg}\right]^{T}\mathbf{F}_g \tag{2.22}$$

and

$$\Delta\phi^{g} = \left[C_{\phi}\right]\left[G_{\phi}^{cg}\right]^{T}\mathbf{F}_g . \tag{2.23}$$

2.2 Alternative Notation of the Robot Modeling

In order to make this report more accessible to broader audience and for the sake of simplicity, we introduce a simplified set of notations for the robot modeling. Both sets of notations will be used extensively throughout the report. Let x denote the EEF position and q the vector of generalized coordinates (typically joint displacements), then the forward kinematics can be described as

$$x = f(q) \tag{2.24}$$

Taking a time derivative of Eq. (2.24) yields

$$\dot{x} = J\dot{q} \tag{2.25}$$

where J is the EEF Jacobian matrix. Taking another time derivative, we obtain

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q} \tag{2.26}$$

The robot dynamics equation can be written as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J^T F_e = \tau \tag{2.27}$$

where $\tau \in \mathcal{R}^n$ is a generalized joint torque vector, $M(q) \in \mathcal{R}^{n\times n}$ a symmetric and positive definite joint inertia matrix, $C(q,\dot{q})\dot{q} \in \mathcal{R}^n$ the Coriolis/centripetal torque, $g(q) \in \mathcal{R}^n$ the torque due to gravity, and $F_e \in \mathcal{R}^m$ the contact force exerted by the EEF.

It is sometimes more convenient to construct a robot controller if the dynamics equation is expressed in the task space. Substituting $\ddot{q}$ from (2.26) in (2.27) yields the dynamics equation in the task space

$$F_c = \Lambda(q)\ddot{x} + \mu(q,\dot{q}) + p(q) + F_e \tag{2.28}$$

where $F_c \in \mathcal{R}^m$ is a task space command force applied to the EEF. $\Lambda(q) \in \mathcal{R}^{m\times m}$ is the operational space kinetic energy matrix given by

$$\Lambda(q) = \left[JM^{-1}J^T \right]^{-1} \tag{2.29}$$

$\mu(q,\dot{q}) \in \mathcal{R}^m$, representing the centripetal and Coriolis forces acting at the EEF, is defined as

$$\mu(q,\dot{q}) = \Lambda\left[JM^{-1}C\dot{q} - \dot{J}\dot{q}\right] \tag{2.30}$$

Finally, $p(q)$ is the gravity forces acting at the EEF and given by

$$p(q) = \Lambda JM^{-1}g \tag{2.31}$$

For clarity and notational convenience, the dependence of J, M, C, $\dot{J}$, and g on q and $\dot{q}$ has not been written explicitly.

In addition, since most manipulators are composed exclusively of revolute joints and thus the joint angles usually become the generalized variables, we also use θ and q interchangeably.

2.3 Control of Manipulators

For robots to be of any use, they must be *controlled* to follow a prescribed path or exert a certain amount of force to the environment. This section reviews many control methods that are currently in use. In a broadest sense, one can group the robot controls into two categories: position and compliant controls. Position control only concerns itself with controlling the positions of the joints (and in turn the EEF position) while compliant control also provides the ability to control forces. The material on control of robotic manipulators is so rich that it can easily fill an entire book. This section is merely meant to provide the reader with a brief introduction to robotic controls and to familiarize the reader with some notations that will be used throughout the report.

2.3.1 Position Control

Position control involves only controlling the motion (position, velocity, and acceleration) of a manipulator, i.e. determining a set of commanded signals that will move the robot along the desired motion trajectory.

2.3.1.1 Independent Joint Control

The simplest of any robotic control method and the most commonly used in industrial robots is *independent joint control*. Joint control does not take into account the dynamics of the robot links. It treats each joint as an independent, uncoupled system and usually employs classical controllers such as PD or PID. The reason that justifies the use of joint control in most robots is the high speed ratios of the gear trains in actuators. These high ratios significantly diminish the effects of the system link inertia. For example, a ratio of 100 would reduce the link inertial effect by 10,000 times. So the robot control problem essentially becomes the problem of controlling the actuators.

Let $q_d(t)$ denote a desired trajectory. We can the define a *tracking error* as

$$e(t) = q_d(t) - q(t) \tag{2.32}$$

For the PID joint controller, the controlled torque is determined by

$$\tau_c = \ddot{q}_d + K_v \dot{e} + K_p e + K_i \varepsilon \tag{2.33}$$

where K_v, K_p, and K_i are the control gains and $e = \dot{\varepsilon}$.

2.3.1.2 Computed-Torque Control

This is also known as *feedback linearization* control. This control method basically tries to linearize the system using the nonlinear dynamic model of the robot. Consider Eq. (2.27) but without the EEF force.

$$M(q)\ddot{q} + N(q, \dot{q}) = \tau \tag{2.34}$$

where $N(q, \dot{q}) \equiv C(q, \dot{q})\dot{q} + g(q)$ represents the nonlinear term.

Solving for $\ddot{q}$ in Eq. (2.34) yields

$$\ddot{e} = \ddot{q}_d + M^{-1}(N - \tau) \tag{2.35}$$

Defining the control input function

$$u = \ddot{q}_d + M^{-1}\left(N - \tau\right), \tag{2.36}$$

we may write the *tracking error dynamics* as

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u \,. \tag{2.37}$$

The controlled torque then can be computed from Eq. (2.36) as

$$\tau_c = M\left(\ddot{q}_d - u\right) + N \tag{2.38}$$

where the output-loop signal $u(t)$ can be chosen using many approaches. For example, if PID outer-loop design is used then

$$u = -K_v\dot{e} - K_p e - K_I\varepsilon \tag{2.39}$$

where $\dot{\varepsilon} = e$. Then, the controlled torque becomes

$$\tau_c = M\left(\ddot{q}_d + K_v\dot{e} + K_p e + K_I\varepsilon\right) + N \,. \tag{2.40}$$

2.3.1.3 Cartesian-Based Control

In the control schemes we have discussed so far in the previous two sections, we assumed that the desired trajectory was available in terms of time histories of joint positions, velocities, and accelerations. We then designed *joint-based control* schemes from these desired joint inputs. Very often though, we would like the robot EEF to follow a prescribed path in the Cartesian coordinates. This requires that the desired EEF trajectory be converted to the desired joint trajectory in the joint-based control schemes. The trajectory conversion is performed by the inverse kinematics.

An alternative is to design control schemes that are based on the errors in Cartesian space. This eliminates the need for the inverse kinematics solutions, which often are complicated. Let's define the *Cartesian error*

$$e_x(t) = x_d(t) - x(t) \tag{2.41}$$

where $x_d(t)$ is the desired Cartesian trajectory and $x(t)$ the actual EEF Cartesian position. With similar derivations to those of the computed-torque control described in the previous section, for non redundant robots, we obtain

$$\tau_c = MJ^{-1}\left(\ddot{x}_d - \dot{J}\dot{q} - u\right) + N \tag{2.42}$$

which is called the *Cartesian computed-torque control law*. For PD control,

$$\tau_c = MJ^{-1}\left(\ddot{x}_d + K_v\dot{e}_x + K_p e_x - \dot{J}\dot{q}\right) + N\ . \tag{2.43}$$

Clearly, the Jacobian inverse does not exist in redundant robots, so the control law Eq. (2.42) does not work. Control of redundant robots is much more involved and is part of the research efforts in this report and therefore will be discussed in later chapters.

2.3.2 Force Control

In many real-world applications where the robot needs to interact with the environment, not only must the position of the EEF be regulated but also the interaction force. This section only introduces briefly two of the most widely accepted force control methods now prevalent in the robotics community. Because force control is an integral part of this research, it will also be discussed in more detail later.

2.3.2.1 Hybrid Position/Force Control

The hybrid position/force control method [Raibert and Craig, 1981] is based on the conjecture that when the end-effector is in contact with the environment, the Cartesian space of the end-effector coordinates can be naturally decomposed into a 'position subspace' and a 'force subspace.' The position/force control problem is then separated into two problems, one which tracks a position trajectory in the position subspace and the second which is to track an independent force trajectory in the force subspace.

The hybrid position/force control is intuitive and simple to implement. However, some tasks such as grinding or deburring cannot be decomposed into position and force subspaces because large amounts of friction along the cutting surface create a constraint that is neither a force or a motion constraint. In other

words, these tasks require the control of force and position along the same direction at the same time.

2.3.2.2 Impedance Control

Impedance control was first introduced by Hogan [1985]. Unlike the hybrid position/force control, it attempts to regulate the dynamic behavior of the end-effector when in contact, i.e. to maintain a desired dynamic relationship between the end-effector's position and the contact force. By selecting proper desired impedance parameters, one can try to compromise the tracking ability of both position and force in the same direction at the same time. However, unlike hybrid position/force control, impedance control does not explicitly track the force trajectory. Thus, one cannot expect to achieve a perfect force tracking even in an obvious "force subspace."

2.4 Summary

This chapter reviews the background materials necessary for understanding the modeling and control of robot manipulators. First, the kinematic, dynamic, and compliance models of manipulators were covered. The development of these models is based on Kinematic Influence Coefficients (KIC). The use of KIC separates the time states from the models, which leads to a simple, clean, and relatively easy to follow notation. The separation of the time states also helps with the development of performance criteria, which will be discussed in the next chapter.

The robot control (position and force) was briefly discussed. Simple position control methods such as independent joint control, computed-torque control, and Cartesian control were introduced. Force control was broadly categorized into two approaches; i.e., hybrid and impedance control. More detailed discussion on force control will be presented in Chapter 7.

CHAPTER 3

REDUNDANCY RESOLUTION AND PERFORMANCE CRITERIA

This chapter presents two tools that together can be used to solve the inverse kinematics problem for redundant manipulators. One of them is Redundancy Resolution Techniques (RRTs) and the other is performance criteria. RRTs are used to search the null space for a given EEF position. The search direction and therefore the solution are determined by some evaluation of chosen performance criteria. Both RRT and performance criteria are built upon the mathematical foundations of manipulator modeling reviewed in the previous chapter. Together, they represent a major component of criteria-based decision making.

3.1 Redundancy Resolution Techniques

This section reviews in detail many of the popular RRTs that are currently in use and also introduces an improvement on the technique employed at UTRRG.

3.1.1 Generalized Inverses

Since the Jacobian of a redundant robot is not square, it cannot be inverted. One way to solve the problem is to use a generalized inverse instead. Using a generalized inverse, the solution to the equation $\dot{x} = J\dot{\theta}$ becomes

$$\dot{\theta} = J^{\#}\dot{x} \tag{3.1}$$

where $J^{\#}$ is a generalized inverse of the Jacobian matrix. One of the more popular generalized inverses is the pseudoinverse (or Moore-Penrose inverse) which for a redundant robot can be defined as

$$J^{\dagger} = J^{T}\left(JJ^{T}\right)^{-1} \tag{3.2}$$

The pseudoinverse solution is known to minimize the two-norm of the joint velocity vector. This solution was originally thought to drive the robot from singularity configurations because singularities are often accompanied by very high demands on the joint velocities. Unfortunately, it has been shown that the pseudoinverse often leads the robot into singularities [Carignan, 1991].

Another generalized inverse used in solving redundancy is the inertia-weighted pseudoinverse, which results in the minimization of the manipulator's kinetic energy [Whitney, 1969]. Let M denote the manipulator's inertia matrix, then the inertia-weighted pseudoinverse is defined as

$$J_I^{\dagger} = M^{-1}J^{T}\left(JM^{-1}J^{T}\right)^{-1} \tag{3.3}$$

The inertia-weighted pseudoinverse is also the only generalized inverse that is dynamically consistent, i.e. the joint torques in its associated null space do not produce any EEF acceleration [Khatib, 1990]. Compared with the pseudoinverse, the calculation of the inertia matrix, however, is computationally expensive, especially for manipulators with high degrees of freedom. In addition, using generalized inverses does not allow any performance criteria to be included to guide the solution to a desired performance gain.

3.1.2 Configuration Control

The configuration control method proposed by Seraji [1989] augments the manipulator forward kinematics with a set of kinematic functions in Cartesian or joint space that reflects the desired additional task. Let $Y = f(\theta)$ be the forward kinematic model of the robot which maps the $n \times 1$ joint displacement vector θ to the $m \times 1$ end-effector coordinate vector Y. Let $\Phi = h(\theta)$ define a set of r (= n-m) kinematic functions. The augmented kinematic model is then given by

$$X = \begin{pmatrix} Y \\ \Phi \end{pmatrix} = \begin{pmatrix} f(\theta) \\ h(\theta) \end{pmatrix} \tag{3.4}$$

where X is the $n \times 1$ configuration vector. The user can then set up the desired additional task by imposing the constraint $\Phi(t) = \Phi_d(t)$, where $\Phi_d(t)$ is the user-specified desired time variation of Φ. The configuration control problem is then to ensure that the configuration vector X tracks the desired trajectory $X_d(t) = \begin{pmatrix} Y_d(t) \\ \Phi_d(t) \end{pmatrix}$ using either a kinematic or dynamic control law.

In a special case where the desired addition task is to optimize an objective function, this method is called the extended Jacobian method, introduced by Baillieul *et al.* [1984]. By defining $\Phi(\theta) = N_e \frac{\partial g}{\partial \theta}$, where N_e is an $r \times n$ matrix that spans the null space of the Jacobian and $g(\theta)$ is the scalar kinematic objective function to be optimized, we can see that $\Phi = 0$ is a necessary condition for optimality of $g(\theta)$. Therefore, if we define the desired trajectory as $\Phi_d(t) \equiv 0$ and use the configuration control to track $X_d(t)$, then the kinematic optimization problem can be solved.

The addition of constraints, however, adds algorithmic singularities, in addition to kinematic singularities, and can restrict the inversion. In order to remedy the problem, Seraji and Colbaugh [1990] proposed a singularity-robust configuration control. It basically followed the singularity-robust inverse kinematics [Nakamura and Hanafusa, 1986] by using the *damped least-squares* method.

3.1.3 Gradient Projection Method

The Gradient Projection Method (GPM) has gained increasing popularity among researchers in the last decade. GPM was first introduced by Liegeois

[1977] to utilize the redundancy to avoid mechanical joint limits. Extending the pseudoinverse solution, a general solution to the inverse kinematics problem can be expressed as

$$\dot{\theta} = J^{\dagger}\dot{x} + (I - J^{\dagger}J)z \tag{3.5}$$

where $J^{\dagger}$ is the pseudoinverse of the Jacobian and z is an arbitrary vector. The first term is a least-norm particular solution and the second term is called homogeneous or null-space solution. The null-space solution does not contribute to the end-effector motion yet determines the joint configuration. To optimize a performance index $h(\theta)$, z is chosen to be

$$z = \pm k\nabla h(\theta) \tag{3.6}$$

where k is a positive real number and $\nabla h(\theta)$ the gradient of $h(\theta)$. A positive sign in Eq. (3.6) indicates that the criterion is to be maximized; a negative sign indicates minimization. From Eq. (3.5), for maximization (the positive sign) the time derivative of the performance index is given by

$$\frac{dh(\theta)}{dt} = \left[\nabla h(\theta)\right]\{J^{\dagger}\dot{x} + (I - J^{\dagger}J)z\} \tag{3.7}$$

The choice of z from Eq. (3.6) has the effect of adding the nonnegative term

$$k\left[\nabla h(\theta)\right](I - J^{\dagger}J)\left[\nabla h(\theta)\right]^{T} \tag{3.8}$$

to the value of $dh(\theta)/dt$, hopefully increasing the value of $h(\theta)$.

With this approach, there is no guarantee that the nonnegative term in Eq. (3.8) will dominate a potential negative effect of the term $\left[\nabla h(\theta)\right]J^{\dagger}\dot{x}$ unless of course the magnitude of k is large enough. If it is too large, however, it can cause unacceptably high joint velocities.

It is worth noting that GPM allows multiple criteria to be incorporated. If multiple criteria are to be optimized, then a single objective can be formed by combining these individual criteria. The weighted sum is by far the most popular method of combining the criteria.

3.1.4 Compromise Solutions

Realizing that MRRP is indeed a *Multiple Criteria Decision Making* (MCDM) problem, Cetin and Tesar [1999] applied decision theory to solving MRRP with a concept of *Compromise Solution (CS)*. CS is a concept in goal programming or goal setting in multiple objective optimizations where even though the solution is not truly optimal (a truly optimal solution implies that every objective is optimized, which is usually unattainable due to conflicts), the decision maker is still satisfied with the solution with respect to his/her goals. Instead of trying to optimize a performance index, which is generally a linear combination of performance criteria, CS attempts to find a solution that makes most, if not all, of the criteria within acceptable levels. Applying the goal programming to MRRP, Cetin introduced a concept of maximum allowable variation (MAV) that addresses the criteria normalization issue [Bevill and Tesar, 1990]. A maximum allowable variation of a criterion is a percent deviation from a target (by default, the target is the optimal value for each criterion) with which the user would be satisfied. This method offers the user a luxury to specify his goals and allows the user to be actively involved in the redundancy resolution process. The CS method can be formulated as follows:

$$\begin{aligned} &\text{minimize} \left[\sum_i w_i^p z_i^p \right]^{1/p} + \psi(b^+; b^-) \\ &\text{subject to } h(x) = 0, g(x) \le 0, \\ &z_j + b_j^+ - b_j^- \le M_j \\ &b_j^+, b_j^- \ge 0, j = 1, \ldots, q \end{aligned} \tag{3.9}$$

where w_i is the weight vector that designates the importance of the ith decision variable with respect to others; p is the order of the l_p-norm; z_i is the percent deviation from a target value of the ith decision variable; b_j^+ and b_j^- are the positive and negative deviations, respectively; $\psi(b^+; b^-)$ is a penalty function that penalizes

any positive deviations in terms of increasing the objective value and encourages any negative deviations; $h(x)$ is a set of the trajectory constraints; $g(x)$ represents other constraints; and M_j is the maximum allowable variation for jth criterion. In summary, CS tries to minimize the l_p-norm of the weighted sum of the percent deviations of all criteria while satisfying the trajectory constraints and attempting to keep each deviation under its MAV. Implementation of this method utilizes Smith and Lasdon's [1992] Generalized Reduced Gradient (GRG) nonlinear optimization algorithm.

By means of maximum allowable variations, CS eliminates the need of criteria normalization as well as offers a systematic procedure to cope with conflicts. Nonetheless, it does have some drawbacks. First of all, the operator needs to enter the target value for each criterion at every instant. Unless the operator is very knowledgeable about the robot and all criteria, it is a difficult chore. Second and more important, while the concept of MAV looks attractive, it does not improve the operator's physical awareness of the task at hand. What does "MAV of 30% for the dexterity criterion" mean to the task? Which one is better, keeping the variation of the actuator coupling criterion under 15% or the stiffness criterion under 20%?

3.1.5 Direct Search Method: Generate Options

Hooper and Tesar [1994] developed a Direct Search (DS) method for first solving the inverse kinematics problem of non-redundant robots. The method uses the forward position solution to guide the search for the inverse solution. The search begins with an estimated solution, which serves as a base configuration. Then small joint perturbations are introduced to generate new configurations and the *error* is calculated at each of these new configurations. A new base configuration that reduces the error is then chosen. This process continues until the search finds a solution, i.e. the error is zero. The search is direct in the sense that it

does not use any derivatives and hence it is robust, especially at or around mathematical singularities. This method was then extended to solving redundancy resolution with multiple criteria.

The completely generalized direct search method is very computationally expensive. The observed computation rates were as slow as 0.6 Hz in a 7-DOF robot with simple performance criteria [Hooper and Tesar, 1994]. To speed up the computation for real-time applications, the multi-criteria inverse kinematics was divided into two stages. A closed-form solution was used to solve the inverse kinematics problem of a selected 6-DOF substructure to meet the EEF constraints. The remaining joints were then used to generate joint-space options using the direct search. Criteria are then evaluated at each of these options and the options are ranked based on some metrics. The option with the best ranking is then selected. Using a closed-form solution however reduces the generality of the method. Kapoor and Tesar [1996] and Kapoor *et al*. [1998] later developed the *Performance-Based Hybrid Generalized Inverse* by utilizing a generalized inverse for the 6-DOF substructure instead of the closed-form inverse. This enables the DS method to perform in real-time applications and yet maintain an expanded generality. With the same 7-DOF robot, a computation rate of 128 Hz (or 170-times improvement over the direct search technique) was reported. Figure 3-1 depicts the diagram of the Hybrid Generalized Inverse.

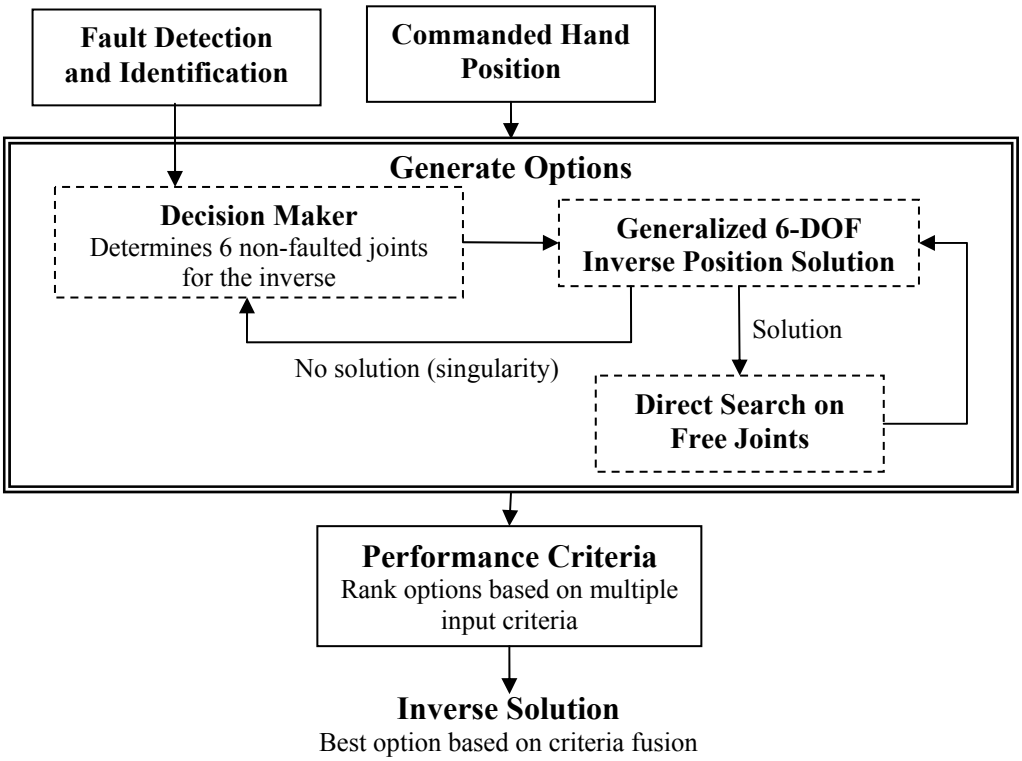


Figure 3-1: Hybrid Formulation for Generalized Inverse [Kapoor and Tesar, 1996].

The 6-DOF generalized inverse is an iterative version of resolved motion rate control scheme by Whitney [1969] and was also proposed by Kelmar and Khosla [1988]. Basically, this technique uses the forward kinematics to compute the EEF error, ε. The Jacobian inverse is then used to calculate the change in joint angles needed to move the EEF to the desired position. The processes are repeated until the EEF error is within the required error tolerance. The whole procedure is detailed in Figure 3-2, where $\left[J_{subset}\right]$ is the Jacobian of the chosen 6-DOF substructure.

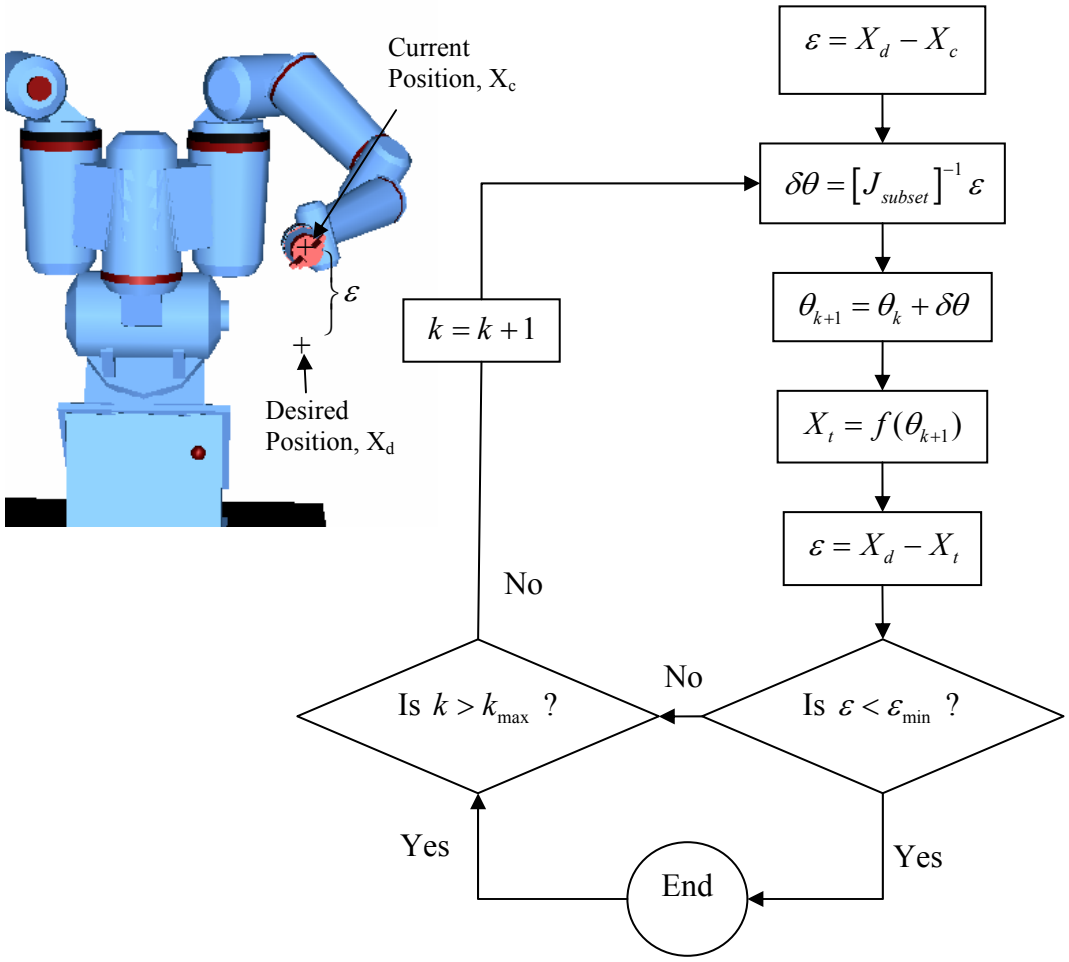


Figure 3-2: Flow chart of the 6-DOF generalized inverse scheme.

The 6-DOF substructure is chosen by the operator to avoid faulted joints or with a strategy that determines a well-conditioned matrix. One drawback of the hybrid inverse versus the pure direct search method is that the 6-DOF generalized inverse relies on inverting the Jacobian matrix, eliminating the robustness of the pure direct search method at or near mathematical singularities. Still, it does not use derivatives of performance criteria when searching for the optimal solution. So it is still direct search in that sense.

When searching for the best solution in the local joint space, one must establish an exploration pattern. Three patterns – namely simple, factorial, and exhaustive – were suggested [Hooper and Tesar, 1994]. Simple exploration involves perturbing only one joint displacement at a time and therefore is the most efficient. Exhaustive pattern, on the other hand, explores in every possible direction and thus generally gives better solution but is very computationally expensive. For a 3-DOF robot, these three patterns correspond to points on the faces, edges, and vertices of a hypercube with the current base point at its center (see Figure 3-3 for example).

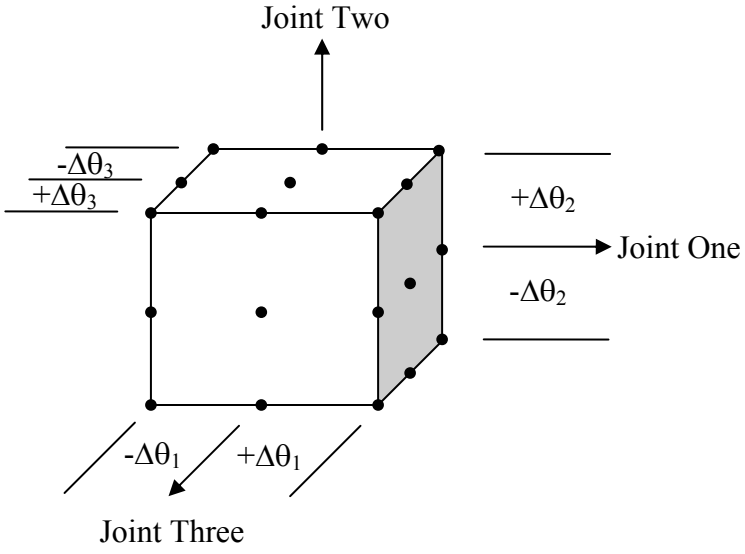


Figure 3-3: A Hypercube Representing the Exploration Patterns for a 3-DOF Robot [Hooper and Tesar, 1994]

One great benefit about this method is that it allows the trade-off between optimality and speed by allowing the user to choose which exploration pattern to be used. To obtain a local optimal solution, one must repeatedly perform the exhaustive search until the value of the objective function does not change. Nevertheless, this process is prohibitively expensive for the objective function that consists of several complex performance criteria. Standard practice at RRG calls

for only a single perturbation step. Thus, because only a finite number of options are generated in one perturbation step, it cannot guarantee any optimality.

Experience has shown that singularities occur when inverting $[J_{subset}]$ even when J is not singular. In other words, the hybrid method introduces *algorithmic singularities*. These algorithmic singularities lead to many invalid solutions when generating options, resulting in both waste of computation time and reduction in search space. Pryor and Tesar [2002] tackled this problem by allowing the operator to lock a set of fewer than *n-m* joints, leaving $[J_{subset}]$ still a non-square matrix. Then a standard pseudoinverse is performed on $[J_{subset}]$ to determine the joint angles that satisfy that EEF requirement. This approach was found to be very successful in reducing the number of algorithmic singularities in demonstrations with a 10-DOF robot. Nonetheless, this method shrinks the volume of search space even further. For example, with a 10-DOF robot, a set of only 27 options can be generated by locking three joints when compared to 81 options by locking four joints for the exhaustive exploration pattern. Furthermore, this method degenerates to the pseudoinverse method when used on robots with one degree of redundancy.

3.1.6 Improved Generate Options

The section proposes two techniques to improve the performance of the existing Generate Options method. The first one is the use of the full Jacobian matrix rather than $[J_{subset}]$ in determining the inverse kinematics solution. This eliminates any possibility of algorithmic singularities associated with the existing GO method. It allows the operator to perturb any number of joints, thus providing the most flexibility when deciding on the size and shape of the search pattern. However, this method requires computing a generalized inverse of the full Jacobian, which is more computationally intensive than inverting $[J_{subset}]$.

The GO algorithm with the full Jacobian can be summarized in the following pseudo code.

$$\begin{aligned} \theta_{base} &= invKin(x_d) \\ \Theta &= generateOptions(\theta_{base}, \Delta\theta) \\ \theta &= pickSolution(\Theta) \end{aligned} \tag{3.10}$$

The symbols in the above pseudo code are designed as follows:

- θ_{base} is the base solution from which the perturbations are performed.
- x_d is the desired EEF position.
- *invKin* is the iterative version of resolved motion rate control using a generalized inverse of the full Jacobian. Depending on the type of the generalized inverse used, the base solution obtained from *invKin* imparts a baseline physical meaning such as minimized joint velocities, minimized instantaneous kinetic energy, or avoiding joint limits.
- Θ is the set of all generated options.
- $\Delta\theta$ is the perturbation size.
- *generateOptions* is an algorithm that applies the user-specified search pattern and perturbs the joints from θ_{base} with $\Delta\theta$ to generate a set of solution options using the method described in the previous section.
- θ is the chosen solution.
- *pickSolution* is any algorithm that picks the best solution from the pool of generated options. For single-criterion redundancy resolution, *pickSolution* is straightforward. It just chooses the solution with the best (maximum or minimum) criterion value. With multiple criteria, a common practice is to use the weighted sum of performance criteria. Task-Based Redundancy Resolution (TBRR), which is a major part of this research and will be discussed in details in the next chapter, can also be used.

The second improvement concerns the ability to find an optimal solution. One problem with the existing Generate Options method with single-step perturbation is that typically, but not always, the larger the perturbation size, the better the solution. However, a large perturbation size can cause high joint velocities and extremely high joint accelerations. To solve this problem, we propose that the current solution should have a bearing on the next solution. That is, it is likely that the chosen solution deviates from the base solution in a similar manner as the previous chosen solution deviates from the previous base solution. Therefore, we add an additional term that is proportional to the previous chosen deviation to the current base solution and generating options around that point instead of the base solution. The improved Generate Options scheme can be expressed as the following pseudo code.

$$
\begin{aligned}
&\delta\theta^0 = 0 \\
&\theta^k_{base} = invKin(x^k_d) \\
&\Theta^k = generateOptions(\theta^k_{base} + \alpha\delta\theta^{k-1}, \Delta\theta) \\
&\theta^k = pickSolution(\Theta^k) \\
&\delta\theta^k = \theta^k - \theta^k_{base}
\end{aligned}
\tag{3.11}
$$

where $\delta\theta^k$ is the joint deviation of the chosen solution at the k-th step and $0 \le \alpha < 1$ is a *momentum* term[5], which specifies how much of an impact the previous joint deviation should have on the current solution. When $\alpha = 0$, the new Generate Options scheme is equivalent to the existing one.

---

[5] The momentum term here serves the purpose similar to the momentum term in the back-propagation algorithm used in the learning of multi-layer perceptrons; hence the name.

3.1.6.1 OSCAR Implementation

This section presents the software implementation of the GO method within the OSCAR framework. The following figure depicts the class diagram of the GO scheme as implemented in OSCAR.

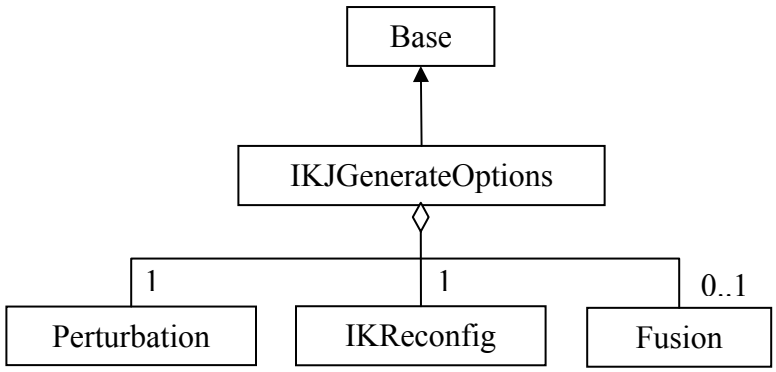


Figure 3-4: Class Diagram of Generate Options.

Base[6] is the base class in OSCAR from which every class is derived. It provides basic functionality common to all OSCAR classes. ***IKJGenerateOptions*** is the class implementing the Generate Options inverse kinematics. As shown in Figure 3-4, it consists of three primary components: ***Perturbation***, ***IKReconfig***, and ***Fusion***. The multiplicity 1's above the first two components indicate that ***IKJGenerateOptions*** contains exactly one object of ***Perturbation*** and ***IKReconfig***. The multiplicity 0..1 signifies that ***IKJGenerateOptions*** can have 0 or 1 instance of ***Fusion*** class, i.e. an object of ***Fusion*** type is optional. ***Perturbation***'s function is to systematically perturb the joints using the three exploration patterns described earlier. The user sets the perturbation size and the search pattern via this class.

---

[6] Actual names of all OSCAR classes are prefixed with RR to differentiate them from third-party classes and to prevent namespace pollution. However, for the sake of brevity and readability, the prefix RR will be omitted here.

IKReconfig is the component that actually performs the inverse kinematics. It takes all the perturbed joint sets and then determines the inverse kinematics solution for each and every set to generate the so called "option." ***Fusion*** then evaluates all the generated options and determines the best solution, i.e. ***Fusion*** performs the *pickSolution* function.

3.1.7 RRT Summary

This section summarizes the advantages and disadvantages of the RRTs reviewed in the previous sections in a tabular format for convenient reference. The summary came from observations based on simulations, experimentations, and analyses at UTRRG.

Inverse Method	Advantages	Disadvantages
Generalized Inverse	• Extremely fast • Smooth joint motion	• No criteria may be implemented • Singularities
Gradient Projection	• Converge quickly to local optimum in self-motion • Smooth joint motion	• Steep gradient may lead to unacceptably high joint velocities • Criteria need to be smooth and differentiable
Configuration Control	• Allows direct control of robot configuration	• Algorithmic singularities
Compromise Solutions	• Criteria need not be normalized • True local optimal solution	• Computationally expensive • Non-smooth joint motion • Criteria need to be smooth and differentiable
Generate Options	• Allows tradeoff between solution quality and speed • Criteria need not be smooth nor differentiable	• Limited by finite set of options • Non-smooth joint motion • Converging to local optimum takes unacceptable amount of time

Table 3-1: Summary of RRTs.

3.1.8 Simulations

This section presents computer simulations that demonstrate the performance improvement of the improved Generate Options scheme introduced in Section 3.1.6 over the current Generate Options method. Here we conduct a simulation on a 4-DOF planar. The 4-DOF planar robot, hereafter referred to as Planar4R, is shown in Figure 3-5. Its properties – including D-H parameters, masses and inertias, etc. – are provided in Appendix A.

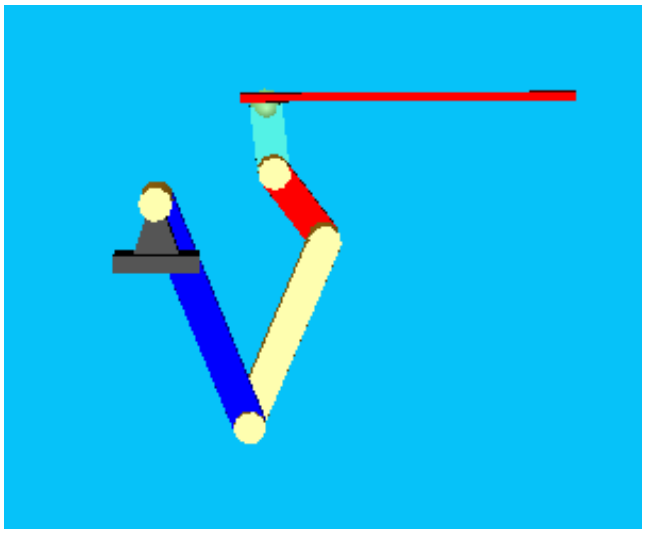


Figure 3-5: Planar4R Robot.

In the simulations, the Planar4R robot follows a straight-line path in the X-direction from points (0.2, 0.1) to (0.6, 0.1). The path is generated using the first-ordered trapezoidal method, i.e. the velocity profile has the shape of a trapezoid. Since only the EEF position, not the orientation, is constrained, the robot has 2 degrees of redundancy. The performance criterion chosen in this case is the Gravity Torque Norm (GTN) criterion. The GTN criterion minimizes the norm of the gravity torque (see Section 3.2.1.3 for detail). With the existing GO, we used $\Delta\theta$ of (a) 0.1 deg and (c) 0.5 deg. We then ran the improved GO with $\Delta\theta$ = 0.1 deg and α = 0.8 (b).

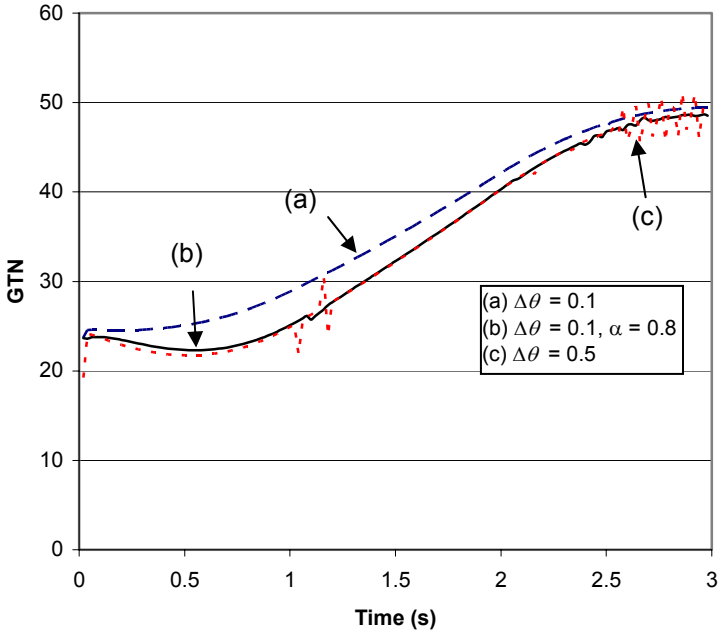


Figure 3-6: GTN criterion values.

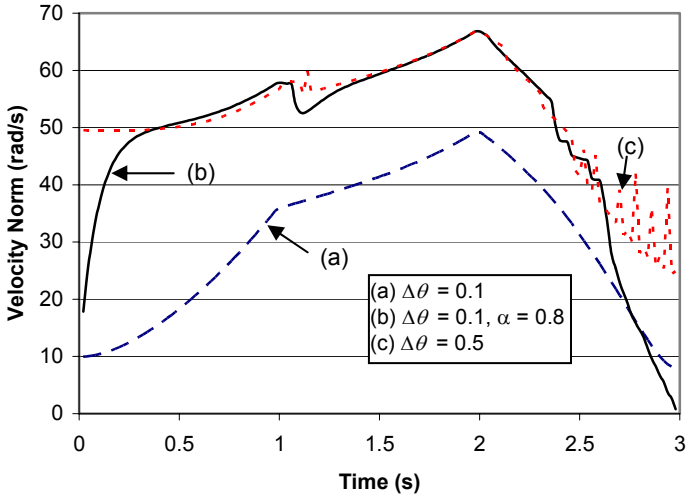


Figure 3-7: Joint velocity norms.

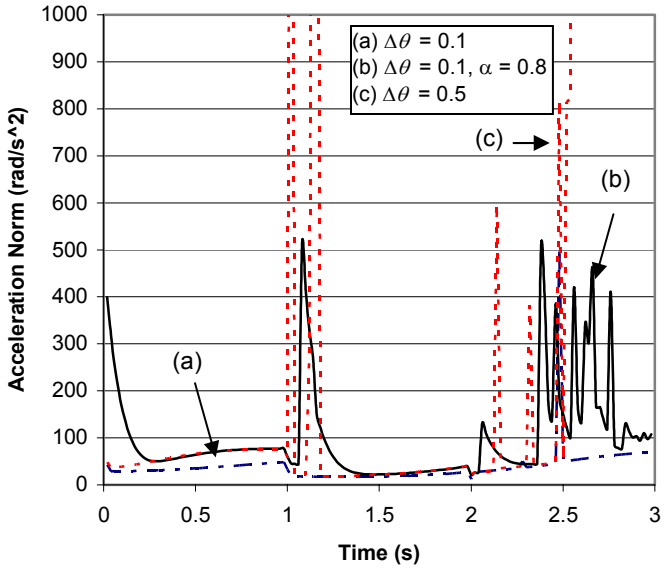


Figure 3-8: Joint acceleration norms.

Figure 3-6 shows the values of the GTN criterion for all three cases. It is clear that cases (b) and (c) yield comparable results which are better than case (a). The criterion value in case (b), however, is smoother than in case (c), especially near the end of the path. Figure 3-7 and Figure 3-8 display the joint velocity norms and the joint acceleration norms[7], respectively, of all three cases. The joint acceleration norm could be roughly used as a measure of trajectory smoothness. The lower the norm, the smoother the solution trajectory is. As can be seen, the better results of cases (b) and (c) come at the expense of the smoothness of joint trajectory compared to case (a). The acceleration in case (c), nonetheless, is much higher than in case (b). In fact, the maximum magnitude for case (c) is around 3,500 rad/s^2 but the plot only shows up to 1,000 rad/s^2 to get a better picture of the other cases.

[7] The velocity and acceleration are approximated by the finite difference method.

The integrals of the criteria values are 5355, 5032, and 5003 for cases (a), (b), and (c), respectively. The integrals of the acceleration norms are 5638, 14907, and 70303 for cases (a), (b), and (c), respectively. Comparing cases (b) and (c), the solution in (b) is only around 0.6% less optimal but 78.8% smoother than (c).

In conclusion, compared with the current GO, the improved GO method with the newly introduced *momentum* can provide better solutions when the same perturbation size is used or a smoother solution trajectory with comparable optimizing capability when using a smaller perturbation size.

3.2 Performance Criteria

When attempting to improve performance of a redundant robotic system for a given task by means of resource allocation, there must be some metrics that are used to determine *optimal* allocation. These metrics are usually called "performance criteria." By definition, performance criteria are real-valued, typically non-negative, functions defined on the joint space to indicate some "states" of a manipulator. A great number of performance criteria have been developed and refined at UTRRG [Bevill and Tesar, 1990] [Van Doren and Tesar, 1992] [Browning and Tesar, 1996] [Hernandez and Tesar, 1996] [Cocca and Tesar, 1999] [Pryor and Tesar, 1999] [Tisius *et al.*, 2004] with many more found in the literature. This section reviews some of these developments, especially those relevant to this research.

3.2.1 Constraint-Based Criteria

Constraint-Based Criteria (CBC) are ones that can be directly associated with physical or mathematical constraints. Violating these constraints will generally result in system failures. It is therefore essential to keep the system from violating these constraints during operation.

3.2.1.1 Criteria for Joint Limit Avoidance

Every joint in a manipulator has its travel limits which cannot be exceeded. Any attempt to move a joint over its limit can potentially damage the robot. By minimizing the joint displacement from its midpoint, joint travel limits can be avoided. The Joint Range Availability (JRA) criterion was defined by Liegeois [1977] as

$$\gamma_{JRA} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\theta_i - \theta_{i,\text{mid}}}{\theta_{i,\max}}\right)^2 \tag{3.12}$$

where θ_i is the joint displacement, $\theta_{i,\text{mid}}$ the displacement at the midpoint of the travel range, and $\theta_{i,\max}$ the displacement at the travel limits. It is clear from the formulation that the criterion value is between 0 and 1 with 0 being the best and 1 being the worst.

Cocca and Tesar [1999] examined the JRA criterion formulation in Eq. (3.12) and noted that when JRA was used with other criteria it was not as effective. Cocca and Tesar later defined the JRA criterion such that its value increases significantly at a configuration near the joint limits.

$$\gamma_{JRA} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\left|\theta_i - \theta_{i,\text{mid}}\right|}{\theta_{i,\max}}\right)^p \tag{3.13}$$

where p is a positive integer that determines the curve of the criterion function as it approaches the joint limit. With a default value of p = 8 as implemented in OSCAR, the criterion function for each joint is plotted as shown in Figure 3-9.

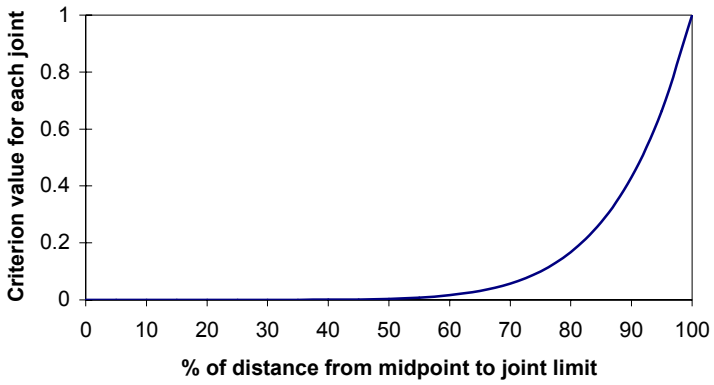


Figure 3-9: Joint Range Availability Criterion

The definitions of JRA criterion above have a benefit of being continuous and differentiable (when p is an even number), which is critical in most optimization techniques. However, it is not a true indicator of joints approaching their limits. Consider a 10-DOF manipulator. Assume one of the joints is at its limit and the others are at their medians. According to Eq. (3.13), we can see that the JRA value would be 0.1. This would seem very acceptable considering the maximum value of JRA is 1. In reality though, this is unacceptable since one of the joints is already at the limit.

A new formulation for JRA has been proposed to remedy this problem and is defined as

$$\gamma_{JRA} = \min_i \left(1 - \frac{\left| \theta_i - \theta_{i,\text{mid}} \right|}{\theta_{i,\max}} \right). \tag{3.14}$$

In this case, the criterion value is still in the range of 0 to 1, but with 0 being the worst and 1 the best. However, when *one or more* of the joints reach their limits, the criterion will approach the value of zero, indicating a bad configuration.

3.2.1.2 Criteria for Velocity Limit Avoidance

Similar to JRA criterion, the joint Velocity Limit Avoidance (VLA) tries to minimize the velocity of each joint or the sum of the velocities of all joints. If the joint speed limit specifications (or their estimates) of the robot are available, then the following formulation can be used to determine VLA criterion.

$$\gamma_{VLA} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|v_i|}{v_{i,\max}}\right)^p \tag{3.15}$$

where v_i is the current joint velocity and $v_{i,\max}$ the maximum velocity allowable for joint i, and p a positive integer. The exponent p serves the same purpose as described for JRA criterion in Section 3.2.1.1. The same arguments in Section 3.2.1.1 can also be made as to why this criterion formulation is not acceptable. Therefore, using a formulation in Eq. (3.14), the Velocity Limit Avoidance (VLA) criterion can be defined as

$$\gamma_{VLA} = \min_i\left(1-\frac{|v_i|}{v_{i,\max}}\right). \tag{3.16}$$

The velocity limit can be avoided by minimizing the norm of the joint velocity vector. The Joint Velocity Minimization (JVM) criterion is defined as

$$\gamma_{JVM} = \dot{\theta}^T\dot{\theta}. \tag{3.17}$$

It is well known that the pseudoinverse solution minimizes the JVM criterion.

Since both of the velocity-related criteria mentioned above are at the velocity level, if the EEF is not in motion, then the criteria will choose the current configuration as the best solution. A manipulator cannot perform self-motion to prepare itself for avoiding the velocity limits.

Browning and Tesar [1996] showed that the ratio between the ∞-norms of the EEF speed vector and the joint velocity is

$$\frac{\|\dot{e}\|_\infty}{\|\dot{\phi}\|_\infty} \geq \left\| \left[G_e^\phi \right] \right\|_\infty^{-1} \tag{3.18}$$

where the ∞-norm gives a vector's absolute maximum element. Another VLA criterion is then defined as the reciprocal of the maximum row sum of $\left[G_\phi^e \right]$.

$$\gamma_{VLA} = \left(\max_i \sum_{j=1}^{m} \left| \left[G_\phi^e \right]_{i,j} \right| \right)^{-1}, \quad (1 \leq i \leq n) \tag{3.19}$$

3.2.1.3 Criteria for Peak Torque Avoidance

A peak torque avoidance criterion can be defined in a similar fashion to joint limit avoidance and velocity limit avoidance criteria.

$$\gamma_{TLA} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{|\tau_i|}{\tau_{i,\max}} \right)^p \tag{3.20}$$

where τ_i and $\tau_{i,\max}$ are the actual torque and the torque limit, respectively, of the i^{th} joint. Although this formulation is simple and straightforward, its use in practice is limited for various reasons. First of all, the torque readings require that torque sensors be present at all actuators. This scenario is not common on many robotic manipulators. Secondly, even with the torque information available, this criterion can only be used to monitor the torque states of the robot but generally cannot be used in redundancy resolution to prevent the robot from exceeding their joint torque limits because most, if not all, redundancy resolution techniques do not work in the force domain.

Another torque limit avoidance criterion that is only configuration-dependent is defined by Browning and Tesar [1996] as

$$\gamma_{TLA} = \left(\left\| \left[I_{\phi\phi}^* \right] \right\|_\infty \left\| \left[G_e^\phi \right] \right\|_\infty \right)^{-1} \tag{3.21}$$

Maximizing this criterion will likely yield a configuration corresponding to a smaller inertia-induced torque demand.

Torque limits can conceivably be avoided by minimizing the joint torques. Torque optimization has long been a topic of research in redundancy resolution. One of the first studies on torque optimization on redundant robots was done by Hollerbach and Suh [1985]. They proposed a method to minimize the two-norm of the instantaneous joint torque vector. The Joint Torque Norm (JTN) criterion is defined as

$$\gamma_{JTN} = \tau^T \tau \,. \tag{3.22}$$

To be able to use this criterion, one must resolve redundancy at the acceleration level. The solution of the torque minimization problem is

$$\ddot{\theta} = J^{\dagger}\left(\ddot{x} - \dot{J}\dot{\theta}\right) - \left[M\left(I - J^{\dagger}J\right)\right]^{\dagger}\left[MJ^{\dagger}\left(\ddot{x} - \dot{J}\dot{\theta}\right) + C + g\right]. \tag{3.23}$$

However, it was shown to become unstable during long trajectories due to excessively high joint accelerations and velocities induced by torque minimization.

To alleviate the instability problem, Nedungadi and Kazerouinian [1989] proposed to minimize inertia inverse weighted torque norm instead of the torque norm itself. This local optimization scheme has an appealing characteristic that its solution "correspond(s) to *one* of the infinite nonoptimal global minimization [of the kinetic energy] variations." The Inertia inverse weighted Torque Norm (ITN) is defined as

$$\gamma_{ITN} = \tau^T M^{-1} \tau \,. \tag{3.24}$$

Like the previous criterion, this one can also only be minimized at the acceleration level with the solution being

$$\ddot{\theta} = J_I^{\dagger}\left(\ddot{x} - \dot{J}\dot{\theta}\right) - \left(I - J_I^{\dagger}J\right)M^{-1}\left(C + g\right). \tag{3.25}$$

Others such as Ma *et al.* [1991] and Kang and Freeman [1992] have proposed other torque optimization schemes. The problem with almost all torque-related criteria is they can only be optimized with acceleration-level redundancy resolution scheme. Also, they cannot be easily combined with other criteria, resulting in specialized solutions.

As noted by Hernandez and Tesar [1996] that "(t)he weight of the structure's links is generally the most dominant loading type, particularly for industrial robots," minimizing the torques due to this gravity load can therefore help in minimizing the total torques, especially at low speed. Also, the gravity torques are easy to calculate and are a function of the joint position. The Gravity Torque Norm (GTN) criterion is simply defined as the Euclidean norm of the gravity torque vector

$$\gamma_{GTN} = \left\| g(\theta) \right\|. \tag{3.26}$$

3.2.1.4 Criteria for Obstacle Avoidance

When a manipulator is utilized in a cluttered environment or in a multi-arm system, the need to avoid obstacles or contacts with other manipulators arises. Based on the KIC model, and the minimum distances and artificial potential fields calculations, Perry and Tesar [1995], Harden and Tesar [1997], and Harden *et al.* [1999] developed, implemented, and evaluated many obstacle avoidance criteria. These criteria assume the knowledge of both the robot(s) and the obstacles.

The obstacle avoidance criteria come in many forms, the simplest of which is the Smallest Minimum Distance (SMD) criterion.

$$\gamma_{SMD} = \min \left| d_{pq} \right| \tag{3.27}$$

where d_{pq} is the minimum distance between link p and obstacle q and calculated for all link/obstacle combinations. This criterion is most likely useful as a trigger to turn obstacle avoidance algorithms on or off.

Another useful minimum distance-based criterion is the Average Minimum distance Reciprocal (AMR) criterion, which is defined as

$$\gamma_{AMR} = \frac{1}{MN} \sum_{p=1}^{M} \sum_{q=1}^{N} \frac{1}{\left| d_{pq} \right|^n}. \tag{3.28}$$

where n serves the same purpose as the variable p described for JRA in Eq. (3.13), M and N are the number of manipulator links and workspace obstacles, respectively. Note that the criterion value is neither dependent on the number of links nor the number of obstacles. Also, dangerous situations are accentuated by the reciprocal format.

We can generate artificial EEF forces and torques from the minimum distance calculations and the current joint configuration. These artificial EEF forces and torques represent the direction and magnitude in which the EEF should be 'pushed' to avoid obstacles. The norms of the artificial EEF forces and torques can be formulated as

$$\gamma_{AEF} = \left\| F_{ee} \right\|_2 = \sqrt{\left(F_{ee}^x \right)^2 + \left(F_{ee}^y \right)^2 + \left(F_{ee}^z \right)^2} \tag{3.29}$$

and

$$\gamma_{AET} = \left\| \tau_{ee} \right\|_2 = \sqrt{\left(\tau_{ee}^x \right)^2 + \left(\tau_{ee}^y \right)^2 + \left(\tau_{ee}^z \right)^2} \ . \tag{3.30}$$

3.2.1.5 Criteria for Mathematical Singularity Avoidance

Physically, at singularities, a manipulator loses one or more degrees of freedom. The robot may not be able to move along the desired direction. Mathematically, it means that the Jacobian is no longer full-ranked and its inverse no longer exists. This is important because many robot controllers and inverse kinematics algorithms rely on the existence of the Jacobian inverse. When the inverse does not exist, most robot controllers fail. Although there are some inverse kinematics methods that can deal with singularities such as the Direct Search method [Hooper and Tesar, 1994] and Partially Exact Least Squares method [Pholsiri *et al*., 2002], singularities should still be avoided.

Since singularities are associated with the Jacobian losing its rank, the minimum singular value of the Jacobian is a good indicator of nearness to a

singularity since it becomes zero at singularities. Van Doren and Tesar [1992] proposed the minimum singular value as the singularity detection (SNG) criterion.

$$\gamma_{SNG} = \sigma_{\min} \tag{3.31}$$

The determinant of a matrix is easily calculated and can be used as another method to measure the relative health of the matrix. Measure of Transmissibility (MOT)[8] defined below is therefore another popular measure of the proximity of the robot configuration to a singularity.

$$\gamma_{MOT} = \sqrt{\det(JJ^T)} \tag{3.32}$$

Singularity can also be detected from the inverse kinematics algorithm itself. Many of the inverse kinematics algorithms including ones currently implemented in OSCAR use inverting a Jacobian-based matrix (maybe the Jacobian itself, $\left[J_{subset}\right]$, $\left[JJ^T\right]$, etc.). An ill-conditioned matrix is prone to numerical errors when inverted. Therefore it is in the best interest to keep this Jacobian-based matrix healthy. The determinant of this matrix can be used as a singularity detection criterion to prevent mathematical failure in inverse kinematics routines.

3.2.2 Operational Goal-Based Criteria (OGBC)

OGBC are not associated with the constraints of the manipulator and thus their physical meanings are not as well understood.

3.2.2.1 Criteria for Dexterity

Dexterity refers to the ability to move or manipulate the EEF. Dexterity is closely related to singularity avoidance. Because, at singularity, the robot loses the ability to move the EEF arbitrarily, dexterity is often used to refer to the ability to

---

[8] It is also known as Measure of Manipulability (MOM) in the literature [Yoshikawa, 1984].

change the position and orientation of the EEF equally well in all directions. The condition number of the Jacobian can be used as a dexterity (DEX) criterion [Van Doren and Tesar, 1992] [Browning and Tesar, 1996]. For use as a performance criterion, the reciprocal of the condition number is better suited than the condition number itself because the reciprocal is bounded in the range between 0 and 1.

$$\gamma_{DEX} = \frac{\sigma_{\min}}{\sigma_{\max}} \tag{3.33}$$

When the dexterity criterion has a value of 0, it means that the robot is at singularity and loses its dexterity. A value of 1 indicates that the robot has the full dexterity and can move its EEF equally well in any direction. A configuration with the dexterity criterion value of 1 is said to be isotropic. From the matrix transformation point of view, the Jacobian of the robot in an isotropic configuration is perfectly conditioned. The robot in an isotropic configuration may move in any direction without exaggerating any errors.

Another dexterity-related criterion is MOT defined in Eq. (3.32). MOT is proportional to the volume of the manipulability ellipsoid. The manipulability ellipsoid represents the set of all EEF velocities that are realizable by the joint velocity $\dot{\theta}$ such that $\left\|\dot{\theta}\right\| \le 1$ ($\|\cdot\|$ is the Euclidian norm). A higher value of MOT – thus a larger volume of the manipulability ellipsoid – means that a greater number of EEF speeds are realizable.

Taking into account the arm dynamics, Yoshikawa [1985] extended the original concept of manipulability and proposed the Dynamic Manipulability Measure (DMM) defined as

$$\gamma_{DMM} = \sqrt{\det\left[J\left(M^T M\right)^{-1} J^T \right]} \tag{3.34}$$

where M is the arm inertia matrix. Like MOT, DMM is proportional to the dynamic manipulability ellipsoid, which roughly represents the set of all EEF accelerations that are realizable by the joint torque τ such that $\|\tau\| \le 1$.

Chiacchio and Concilio [1998] later reformulated the dynamic manipulability ellipsoid to account for the gravity effect more correctly. In the general case where the limits of the joint torques are not equal, it is useful to scale them. If we assume that torque limits are independent of joint velocities and symmetric such that

$$-\tau_i^{limit} \le \tau_i \le +\tau_i^{limit}, \qquad i = 1,\ldots,n, \tag{3.35}$$

then the normalized joint torques can be written as

$$\tilde{\tau} = L^{-1}\tau, \tag{3.36}$$

where $L = \mathrm{diag}\left(\tau_1^{limit}, \tau_2^{limit}, \ldots, \tau_n^{limit}\right)$ is the scaling matrix.

The dynamic manipulability ellipsoid can be expressed as

$$\left(\ddot{x} - \ddot{x}_g\right)^T \left(JQ^{-1}J^T\right)^{-1} \left(\ddot{x} - \ddot{x}_g\right) \le 1 \tag{3.37}$$

where $\ddot{x}$ is the EEF acceleration in the task space; $\ddot{x}_g = -JM^{-1}g(\theta)$; and $Q = ML^{-2}M$ is an $n \times n$ positive definite matrix. Neglecting the gravity term, the Scaled Dynamic Manipulability measure (SDM) can be defined as

$$\sqrt{\det\left[JQ^{-1}J^T\right]} \tag{3.38}$$

SDM is more useful than DMM since it incorporates the joint torque limits into its formulation.

3.2.2.2 Criteria for Speed of Operation

The linear relationship between the EEF speed and the joint velocity can be expressed as

$$\dot{x} = J\dot{\theta} \tag{3.39}$$

Applying the Euclidean vector norm and Frobenius matrix norm to the above equation, Thomas *et al*. [1985] used Rayleigh's quotient to quantify the transmission of the joint velocities to the EEF velocity and determined that the Euclidean norm of the EEF speed is bounded by

$$\sqrt{\lambda_{\min}}\left\|\dot{\theta}\right\| \le \left\|\dot{x}\right\| \le \sqrt{\lambda_{\max}}\left\|\dot{\theta}\right\|. \tag{3.40}$$

where $\lambda_{\min}$ and $\lambda_{\min}$ are the minimum and maximum eigenvalues of JJ^T.

Since the square root of the eigenvalue JJ^T is equivalent to the singular value of the Jacobian, Eq. (3.40) can be rewritten as [Browning and Tesar, 1996]

$$\sigma_{\min} \le \frac{\left\|\dot{x}\right\|}{\left\|\dot{\theta}\right\|} \le \sigma_{\max} \tag{3.41}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the minimum and maximum singular values of the Jacobian, respectively. From Eq. (3.41), it means that for a fixed set of the joint speed, the norm of the EEF speed is bounded from below by $\sigma_{\min}$. The higher the value of $\sigma_{\min}$ is, the larger the lower bound of $\left\|\dot{x}\right\|$ will be. Therefore, the EEF speed can be increased by maximizing $\sigma_{\min}$. The Generalized EEF Speed (GES) criterion can then be defined as

$$\gamma_{GES} = \sigma_{\min}. \tag{3.42}$$

GES criterion is suited in general cases where no direction of high speed is specified. In some cases where high speed is only required in a certain direction, the task-dependent Velocity Transmission Ratio (VTR) criterion may be more useful. Using the *velocity ellipsoid*, Chiu [1987] defined the VTR criterion as

$$\gamma_{VTR} = \left(u^T\left(JJ^T\right)^{-1}u\right)^{-\frac{1}{2}} \tag{3.43}$$

where *u* is the unit vector in the direction of interest. Maximizing VTR will minimize the joint velocity required to produce a given EEF speed in the *u*

direction or maximize the EEF speed in the desired direction for a given joint velocity.

3.2.2.3 Criteria for Load Carrying Capacity

Forces in the joint space and task space are mapped via the Jacobian through the relation

$$\tau = J^T f \tag{3.44}$$

Again, applying Rayleigh's quotient yields the upper and lower bounds of the Euclidean norm of the EEF force as [Thomas *et al*., 1985]

$$\frac{\|\tau\|}{\sqrt{\lambda_{\max}}} \le \|f\| \le \frac{\|\tau\|}{\sqrt{\lambda_{\min}}}. \tag{3.45}$$

Using the singular values instead of the eigenvalues results in the ratio between the Euclidean norms of the EEF force and the joint torque [Browning and Tesar, 1996]

$$\frac{1}{\sigma_{\max}} \le \frac{\|f\|}{\|\tau\|} \le \frac{1}{\sigma_{\min}}. \tag{3.46}$$

For a fixed joint torque vector, the norm of the EEF force is bounded from below by $\sigma_{\max}^{-1}$. Therefore, the EEF force can be increased by maximizing $\sigma_{\max}^{-1}$ (in other words, minimizing $\sigma_{\max}$). We can then define the Generalized EEF Force (GEF) criterion as

$$\gamma_{GEF} = \sigma_{\max} \tag{3.47}$$

Like GES, GEF is more suited for general cases where task information is not readily available. However, if the desired direction of high EEF force is

known, then the task-dependent Force Transmission Ratio (FTR)[9] criterion may be more valuable. The FTR criterion is formulated as [Chiu, 1987]

$$\gamma_{FTR} = \left(u^T \left(JJ^T \right) u \right)^{-\frac{1}{2}}. \tag{3.48}$$

Again, u is the unit vector in the direction of interest. Maximizing FTR will increase the EEF force capability in the desired direction. Looking at the formulations of VTR and FTR, it can be obviously concluded that they are not by any means independent. One cannot expect to achieve arbitrary values along the same direction of both criteria simultaneously. In fact, when u coincides with one of the principal axes of the ellipsoid, γ_{VTR} and γ_{FTR} are the reciprocals of each other.

3.2.2.4 Criteria for Manipulator Precision

Manipulator's joints are expected to have some amount of error, including position sensor error (encoder resolution or noise), control error, and deflection due to joint compliance. These joint errors are propagated through the links and to the EEF. Minimizing the effect of this error propagation is essential in applications requiring precise manipulation [Hill and Tesar, 1997].

Similarly to the mapping between the joint velocities and the EEF velocity in Eq. (3.39), an infinitesimal error in the EEF position can be mapped from the joint errors through

$$\Delta x = J \Delta \theta . \tag{3.49}$$

The Euclidean norm of the EEF error is bounded above and below by

$$\sigma_{\min} \le \frac{\|\Delta x\|}{\|\Delta \theta\|} \le \sigma_{\max} . \tag{3.50}$$

---

[9] The Force Transmission Ratio is also known as Mechanical Advantage in the literature on mechanisms.

Therefore, if $\sigma_{\max}$ is minimized, then the overall effect that the joint errors have on the EEF may be diminished.

Again, if the precision is desired only in a certain direction, then the VTR criterion defined in Eq. (3.43) may be used. However, minimizing the VTR criterion is needed in this case. We can then see that one cannot achieve both minimum EEF error and maximum EEF speed in the same direction at the same time.

More often than not, the links of robots, especially industrial ones, are assumed to be rigid. This assumption simplifies the design of robot controllers. In reality, robots are compliant or flexible, particularly at their joints in which actuators reside. Under load, joint compliance causes joint deflections which are in turn reflected to the EEF. For a precise operation, the EEF deflection due to joint compliance must be compensated for or minimized.

Given the joint compliance and the robot configuration, the stiffness matrix can be calculated by Eq. (2.21). The Generalized EEF Load Stiffness (GLS) quantifies the size of the stiffness matrix using the Euclidean norm of the vector of the eigenvalues of the stiffness matrix [Van Doren and Tesar, 1992] and allows the stiffness of various configurations to be compared. The GLS criterion is defined as

$$\gamma_{GLS} = \left(\sum_{i=1}^{m} \lambda_i^2 \right)^{\frac{1}{2}} \tag{3.51}$$

where λ_i is the i^{th} eigenvalue of the stiffness matrix.

For tasks that demand accuracy in some certain directions, it is possible to place a robot in such a configuration that is stiffer in those directions. Minimizing the unit Directional EEF Load Stiffness (DLS) criterion defined below will lead to a smaller EEF defection due to the load in the direction of the unit vector u [Pryor and Tesar, 1999].

$$\gamma_{DLS} = \left(\sum_{i=1}^{m} \left(\frac{1}{\lambda_i} u^T S_i \right)^2 \right)^{-\frac{1}{2}} \tag{3.52}$$

where λ_i and S_i is the i^{th} eigenvalue and eigenvector of the EEF stiffness matrix defined in Eq. (2.21), respectively.

The potential energy owing to the deflections can also be used as a criterion. If only the deflections due to forces at the EEF are considered, then the EEF Potential Energy (EPE) criterion can be defined as

$$\gamma_{EPE} = \frac{1}{2} \left(f_e^T \left[K \right]^{-1} f_e \right) \tag{3.53}$$

where f_e is the EEF load and $\left[K \right]$ the EEF stiffness matrix defined in Eq. (2.21).

3.2.2.5 Criteria for Energy Minimization

Kinetic energy minimization is one of the early criteria used in redundancy resolution because kinetic energy is directly associated with the power consumed by the system during operation. The kinetic energy of a manipulator is defined as

$$\gamma_{KE} = \frac{1}{2} \dot{\phi}^T \left[I^*_{\phi\phi} \right] \dot{\phi}. \tag{3.54}$$

The inertia-weighted pseudoinverse can be used to solve the inverse kinematics problem while minimizing the instantaneous kinetic energy [Whitney, 1969]. The inertia-weighted pseudoinverse is defined as

$$J_I^+ = \left[I^*_{\phi\phi} \right]^{-1} J^T \left(J \left[I^*_{\phi\phi} \right]^{-1} J^T \right)^{-1}. \tag{3.55}$$

Observing that the inertia matrix is the main contributor in the kinetic energy, Van Doren and Tesar [1992] proposed a task-independent criterion called Inertia matrix Frobenius Norm (IFN):

$$\gamma_{IFN} = \left(\sum_{i=1}^{n} \lambda_i^2 \right)^{\frac{1}{2}} \tag{3.56}$$

By minimizing the above criterion, the kinetic energy *should* be reduced. However, it is possible that the joint velocities need to increase in order to minimize the above criterion. The increase in the joint velocities could have a greater impact on the kinetic energy than the reduction of the inertia matrix and therefore cause the kinetic energy to increase.

Power consumption of manipulators, especially ones with electric motor driven joints, is closely related to the commanded joint torque. By minimizing the joint torque, the power consumption can be reduced. Therefore, all the joint torque optimization criteria discussed in Section 3.2.1.3 can be applied here. Another component of power is the velocity. Therefore, we could also use criteria for minimizing joint velocity presented in Section 3.2.1.2 for this purpose.

3.2.2.6 Other Criteria

One of the main hindrances to utilizing redundant manipulators in an industrial environment is joint drift. Joint drift occurs when the joints do not return to their initial positions while completing a closed path. A motion with no joint drift is said to be *conservative* or *cyclic*. Conservative motions are desired because the initial configuration in general is strategically chosen so that it represents a good configuration. Returning to the good initial configuration is therefore deemed a good practice. Joint Drift Minimization (JDM) criterion is defined as

$$\gamma_{JDM} = \frac{1}{2}\left(\theta - \theta_0\right)^T W \left(\theta - \theta_0\right) \tag{3.57}$$

where θ_0 is the initial joint configuration and W is a positive definite weight matrix. The criterion value of zero indicates a cyclic motion.

Li and Sastry [1987] proposed the concept of *task ellipsoid* as a means to modeling tasks. The shape of a task ellipsoid is determined from the relative force requirements along the task directions. Based on the task ellipsoid concept, Chiu [1988] suggested that the principal axes of the task ellipsoid correspond to the task

directions and that the lengths of the axes be equal to the desired nominal force transmission ratios. The reason FTR is used instead of VTR is because the calculation of FTR does not involve inverting $\left(JJ^T\right)$. Not only does this reduce the computational complexity but also avoids the absence of $\left(JJ^T\right)$ inverse at singularities. We can now define the Force Transmission Ratio Deviation (FTD) criterion as:

$$\gamma_{FTD} = \sum_{i=1}^{m} w_i \left(\frac{T_{f,i}}{T_{d,i}} - 1 \right)^2 \tag{3.58}$$

where w_i is the weighting factor along the task direction u_i, $T_{f,i}$ and $T_{d,i}$ denote the actual and desired force transmission ratios along u_i, respectively. Naturally, we would like to minimize this criterion.

Manipulators are highly coupled systems, i.e. a single joint torque depends on the motion of many other joints. Most robotic manipulators, especially industrial-type, are controlled by Independent Joint Controller (IJC). IJC assumes that each joint is decoupled when in fact they are not. It is therefore desirable to minimize the coupling among joints so that each actuator torque is least dependent on other joints.

Considering only the acceleration-induced joint torques, Van Doren and Tesar [1992] proposed to minimize the Frobenius norm of the matrix that would result if the diagonal elements of the inertia matrix were replaced by zeros. The Dynamic Coupling (DC) criterion is defined as

$$\gamma_{DC} = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \left[\left(1 - \delta_{ij}\right) \left(i_{ij}^*\right)^2 \right] \right)^{\frac{1}{2}} \tag{3.59}$$

where δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ when $i = j$ and 0 otherwise) and i_{ij}^* is the element of the inertia matrix $\left[I_{\phi\phi}^* \right]$ on the i-th row and j-th column. It should be

noted here that while it is believed that minimizing inertia coupling can improve control performance, no proof has ever been made that quantifies such improvement.

3.2.3 Performance Criteria Summary

For convenience, some select performance criteria that have been discussed in this chapter are summarized in the following table along with their formulations and reference equations, their physical interpretation, and their perceived significance to robotic tasks.

Purpose	Criteria and Formulations	Reference Equation	Interpretation	Significance to Tasks
Joint Limit Avoidance	Joint Range Availability (JRA): $\gamma_{JRA} = \min_i \left(1 - \frac{\left\| \theta_i - \theta_{i,\text{mid}} \right\|}{\theta_{i,\max}} \right)$	(3.14)	JRA = 0: One or more joints are at their travel limits. JRA = 1: All joints are at the midpoints of the travel ranges.	It is crucial to keep JRA from approaching 0. However, it is not important to maximize JRA.
Velocity Limit Avoidance	$\gamma_{VLA} = \min_i \left(1 - \frac{\left\| v_i \right\|}{v_{i,\max}} \right)$	(3.16)	VLA = 0: One or more joints move at their speed limits. VLA = 1: All joints are stationary.	It is crucial to keep VLA from approaching 0. However, it is not important to maximize VLA.
Peak Torque Avoidance	Torque Limit Avoidance (TLA): $\gamma_{TLA} = \left(\left\| \left[I^*_{\phi\phi} \right] \right\|_\infty \left\| \left[G^\phi_e \right] \right\|_\infty \right)^{-1}$	(3.21)	Physical interpretation of this criterion is unclear.	Minimizing TLA should lead to reduced torque demands.
	Joint Torque Norm (JTN): $\gamma_{JTN} = \tau^T \tau$	(3.22)	JTN = 0: All joint torques are zero.	Minimizing JTN reduces the torque demands on the joints. However, it requires solving redundancy at the acceleration level and could lead to instability for a long-trajectory task.
	Gravity Torque Norm (GTN): $\gamma_{GTN} = \left\| g(\theta) \right\|$	(3.26)	GTN = 0: All the joint torques due to gravity are zero.	Minimizing GTN tends to minimize the total torques, especially for low-speed operation.
Obstacle Avoidance	Smallest Minimum Distance (SMD): $\gamma_{SMD} = \min \left| d_{pq} \right|$	(3.27)	SMD = 0: One or more links of the robot collide with an obstacle.	It is crucial to keep SMD from approaching 0. SMD is typically used to turn obstacle avoidance algorithm on or off.
	Average Minimum distance Reciprocal (AMR): $\gamma_{AMR} = \frac{1}{MN} \sum_{p=1}^{M} \sum_{q=1}^{N} \frac{1}{\left| d_{pq} \right|^n}$	(3.28)	AMR $\to 0$: All the obstacles are very far from the manipulator. AMR $\to \infty$: The manipulator is approaching one or more obstacles.	It is crucial to keep the AMR value close to 0 (or a relatively small positive threshold)

Purpose	Criteria and Formulations	Reference Equation	Interpretation	Significance to Tasks
Mathematical Singularity Avoidance	Singularity Detection (SNG): $\gamma_{SNG} = \sigma_{\min}$	(3.31)	SNG = 0: Singularity approached.	To avoid mathematical software failure, it is crucial to keep SNG from approaching 0. However, it is not important to maximize SNG.
	Measure of Transmissibility (MOT): $\gamma_{MOT} = \sqrt{\det(JJ^T)}$	(3.32)	MOT = 0: Singularity approached.	For the purpose of avoiding mathematical software failure, it is crucial to keep MOT from approaching 0. However, it is not important to maximize MOT.
Dexterity	Dexterity (DEX): $\gamma_{DEX} = \frac{\sigma_{\min}}{\sigma_{\max}}$	(3.33)	DEX = 0: The robot loses the ability to move in arbitrary direction. DEX = 1: The robot is in the isotropy state, i.e. the robot can move in any direction equally well.	It is desirable to maximize DEX.
	Scaled Dynamic Manipulability measure (SDM): $\sqrt{\det\left[JQ^{-1}J^T\right]}$	(3.38)	SDM = 0: The robot loses the ability to accelerate in an arbitrary direction.	It is desirable to maximize SDM.
Speed of Operation	Generalized EEF Speed (GES): $\gamma_{GES} = \sigma_{\min}$	(3.42)	Since $\|\dot{x}\| \geq \sigma_{\min} \cdot \|\dot{\theta}\|$, the larger the GES value, the larger the EEF velocity norm.	For a general task, it is desirable to *maximize* GES.
	Velocity Transmission Ratio (VTR): $\gamma_{VTR} = \left(u^T \left(JJ^T\right)^{-1} u\right)^{-\frac{1}{2}}$	(3.43)	VTR = 0: The robot loses ability to move in the *u* direction no matter how fast the joints move. $VTR \to \infty$: The robot can easily move in the *u* direction even with small movements from the joints.	It is important to *maximize* VTR in the task direction that requires high speed operation.

Purpose	Criteria and Formulations	Reference Equation	Interpretation	Significance to Tasks
Load Carrying Capacity	Generalized EEF Force (GEF): $\gamma_{GEF} = \sigma_{\max}$	(3.47)	Since $\|f\| \geq \|\tau\| / \sigma_{\max}$, the smaller the GEF value, the larger the EEF force norm.	For a general task, it is desirable to *minimize* GES.
	Force Transmission Ratio (FTR): $\gamma_{FTR} = \left(u^T \left(JJ^T \right) u \right)^{-\frac{1}{2}}$	(3.48)	FTR = 0: The robot loses the ability to exert an EEF force in the *u* direction. FTR $\to \infty$: The robot can exert a large amount of EEF force in the *u* direction even with small torques from the joints.	It is important to *maximize* FTR in the task direction that requires large EEF force.
Manipulator Precision	Velocity Transmission Ratio (VTR): $\gamma_{VTR} = \left(u^T \left(JJ^T \right)^{-1} u \right)^{-\frac{1}{2}}$	(3.43)	VTR = 0: Joint errors do not generate any EEF error in the *u* direction. VTR $\to \infty$: Small joint errors generate a large EEF error in the *u* direction.	It is important to *minimize* VTR in the task direction that requires high precision.
	Generalized EEF Load Stiffness (GLS): $\gamma_{GLS} = \left(\sum_{i=1}^{m} \lambda_i^2 \right)^{\frac{1}{2}}$	(3.51)	The larger the GLS value, the larger the "size" of the stiffness matrix.	For a general task, it is desirable to *maximize* GLS.
	EEF Potential Energy (EPE): $\gamma_{EPE} = \frac{1}{2} \left(f_e^T \left[K \right]^{-1} f_e \right)$	(3.53)	The instantaneous potential energy of the manipulator from elastic deformations due to the EEF load.	For a general task, it is desirable to *minimize* EPE.

Purpose	Criteria and Formulations	Reference Equation	Interpretation	Significance to Tasks
Energy Minimization	Kinetic Energy (KE): $\gamma_{KE} = \frac{1}{2}\dot{\phi}^T \left[I^*_{\phi\phi} \right] \dot{\phi}$	(3.54)	The instantaneous kinetic energy of the manipulator.	For a general task, it is desirable to *minimize* KE.
	$\gamma_{IFN} = \left(\sum_{i=1}^{n} \lambda_i^2 \right)^{\frac{1}{2}}$	(3.56)	The larger the IFN value, the larger the "size" of the inertia matrix.	For a general task, it is desirable to *minimize* IFN.
Cyclic Motion	Joint Drift Minimization (JDM): $\gamma_{JDM} = \frac{1}{2}\left(\theta - \theta_0\right)^T W \left(\theta - \theta_0\right)$	(3.57)	JDM = 0: There is no joint drift, i.e. the motion is conservative or cyclic.	It is desirable to minimize JDM, especially for repetitive tasks.

Table 3-2: Summary of Performance Criteria.

3.2.4 Current Issues with Performance Criteria and RRTs

The previous two sections discussed two main components used to resolve redundancy. RRTs search the null space for an optimal solution based on evaluation of performance criteria.

3.2.4.1 Scaling

Scaling is an inherent problem associated with mathematical modeling of manipulators when translational and rotational quantities are mixed. Bevill and Tesar [1990] demonstrated the impact of scaling on a 7-DOF arm using the condition number as an example. When lengths are expressed in meters and all joint angles are at 45 degrees, the singular values are

$$\begin{bmatrix} 1.9709 & 1.7518 & 1.3687 & 0.5218 & 0.4772 & 0.1525 \end{bmatrix}.$$

Switching the unit of lengths to centimeters for the same configuration yields the singular values of

$$\begin{bmatrix} 113.6 & 99.51 & 26.42 & 1.493 & 0.7795 & 0.5161 \end{bmatrix}.$$

The condition numbers in the two cases are 12.9 and 220.0, respectively. One can clearly see that the translational components dominate the value of the condition number when centimeters are used. From Section 3.2.1.5, we learned that the condition number is one of the indicators used to determine the proximity to singularity. If one is asked to interpret these condition numbers, he would conclude that the robot is close to singularity in the first case and far from it in the second even though both represent the same robot's physical configuration. Many, including Bevill and Tesar [1990] and Gosselin [1990], have suggested solutions to this problem. However, these solutions just shift the arbitrariness from the unit of choice to others such as characteristic length or reference points.

3.2.4.2 Normalization

In multi-criteria redundancy resolution, the weighted sum of the performance criteria is generally employed. A subset of criteria which best describe the task at hand is chosen. They are then weighted according to their relative import and summed into a composite performance index. Because performance criteria are of different functional forms and magnitudes, they must be *normalized* to allow reliable comparison and combination. With normalization, criteria with different numerical ranges are altered to comparable magnitudes so that, when combined, no single criterion can excessively dominate the solution. Effective normalization is not trivial as it involves knowing the normalizing factors (e.g. criteria's maximum values) of all the criteria [Bevill and Tesar, 1990].

Tisius *et al.* [2004] presented an empirical approach to normalizing performance criteria. The manipulator's performance maps are generated empirically via computer simulation by running the robot over the course of a prescribed path. These performance maps can then be used to normalize performance criteria so that they are comparable in magnitudes. This normalization process "allows the user to make valid comparisons, and aids the RRT's ability to prioritize" [Tisius *et al.*, 2004]. This empirical approach also allows RRTs to employ *global normalization* instead of the current use of *local normalization*. Global normalization was shown to be more stable, reflect the physical state of the robot more accurately, and handle conflicts among multiple criteria more effectively.

3.2.4.3 Levels of Redundancy Resolution

From the criteria formulations defined previously, one can see the diversity. Criteria can be functions of the robot's joint displacements, velocities, accelerations, or torques. This does not seem to be a problem until one attempts to combine criteria defined at different levels to solve for an optimal configuration.

Most, if not all, RRTs are designed to resolve redundancy at one particular level. The gradient projection method, for example, solves the inverse kinematics problem at the velocity level and can only deal with criteria defined at the position level. The Compact Quadratic Programming method [Cheng *et al.*, 1992], on the other hand, resolves redundancy at the velocity level with velocity-dependent criteria whereas Direct search or Generate Options works at the position level as far as performance criteria and redundancy resolution are concerned.

3.2.4.4 Task Dependency

Task independency has long been a central theme in the realm of performance criteria and redundancy resolution, especially at UTRRG. It was believed that performance criteria should be task-independent because they would then need to be formulated only once [Hooper and Tesar, 1994]. Also, task-independent criteria are functions of only manipulator configuration and states and are not tied to any specific task. This makes the criteria very useful for the system designer, who cannot foresee all the possible tasks to be performed by the robot.

Nevertheless, we argue here that task-dependent criteria should not be dismissed entirely. It is true that task-independent criteria need merely be formulated once. However, it is also true that task-dependent criteria can be formulated just once. Only in their evaluations, not formulations, will the task dependency come into play. While task-independent criteria may prove invaluable to the system designer, task-dependent criteria usually give more insight to the operator on the task at hand.

3.2.4.5 Couplings and Conflicts among Criteria

A quick glance at the list of performance criteria in Table 3-2 reveals an undeniable fact that most, if not all, of these criteria are coupled. It is therefore not possible to optimize one criterion without affecting another. For example, if we attempt to maximize the JRA criterion, it is possible that this may have an impact

on the VTR criterion. Even though the intention of including JRA in the redundancy resolution process is merely to avoid the joint limits, we may inadvertently increase or decrease the ability of the robot to move in a desired direction. These couplings also make it impossible to completely separate the purposes of these criteria, making the task of choosing criteria for a given task very difficult.

Another ramification that results from the couplings is conflicts among criteria. The best example is the conflict between the speed and force capabilities of the manipulator. Normally, one would like to maximize both of them at the same time. Nevertheless, because of the conflicting nature of these two quantities, it is just physically impossible. A closer look at the formulations of VTR and FTR criteria shows that these two criteria are tightly coupled. As a matter of fact, in some special cases, they are the reciprocals of each other. Another example is maybe even more obvious. It was said that VTR can be used to either increase the EEF speed or the EEF precision. However, while increasing the EEF speed requires that VTR be maximized, improving the EEF precision demands the opposite. These conflicts also lead to a difficulty of choosing appropriate criteria for a given task. The problems of couplings and conflicts among performance criteria are one of the main motivations behind this research.

3.3 Summary

This chapter first reviewed some of the most popular redundancy resolution techniques currently in use and discussed their strengths and weaknesses. These methods included generalized inverses, configuration control, gradient projection, compromise solutions, and generate options.

Also introduced was the *improved* version of the Generate Options method. The improved GO method addressed two major issues that exist in the current GO method. First, it eliminates the algorithmic singularity problem by using all

available joints to generate options. Second, the momentum term introduced in the improved GO helps generate a *smoother* solution trajectory while maintaining approximately the same quality (optimality) of the solution. The simulation with the Planar4R robot showed that the improved GO yielded a solution that was around 0.6% less optimal but 78.8% smoother than a solution from the current GO.

The second part of the chapter examined in detail performance criteria, which are the metrics used to determine the "optimal" solution in the redundancy resolution process. The performance criteria were broadly divided into two categories: constraint-based criteria and operational goal-based criteria. As its name implies, the constraint-based criteria are directly associated with system (physical or mathematical) constraints (e.g. joint limits, obstacles, singularities, etc.). Therefore, the constraint-based criteria in general have clear physical meanings that the user can easily relate to. Operational goal-based criteria, on the other hand, are concerned with the ability of the robot to perform the task better. They were organized in terms of the associations with the high level task-oriented concepts of:

- Dexterity
- Speed of Operation
- Load Carrying Capacity
- Manipulator Precision
- Energy Minimization
- Conservative Motion

Finally, some issues associated with the current performance criteria formulations such as scaling, normalization, task dependency, couplings and conflicts were discussed.

CHAPTER 4

TASK REQUIREMENTS AND ROBOT CAPABILITIES

The previous chapter reviewed two traditional components essential for deploying redundant systems, namely RRTs and performance criteria. So far, no explanation of robotic tasks, which should be integral to every operating scheme, has been mentioned. In this chapter, we will describe the concepts of task requirements, robot capabilities, and task executability. Task requirements will be defined and the formulations for estimating the robot capabilities will be formalized. A new approach to robot capability estimations called the *vector expansion method* will be developed and the results discussed and compared with the conventional ellipsoid method. The proposed formulations for estimating the robot capabilities will serve as an important tool in the Task-Based Redundancy Resolution (TBRR), which will be presented in the next chapter.

4.1 Task Requirements

The previous chapter discusses a selection of performance criteria which are to be optimized when choosing the best configuration for a robot with excess input resources. It should be noted that most of those criteria do not have relevant physical meanings at the task level to which the operator can relate. A typical non-expert operator may not care about the values of manipulability, stiffness, or transmission ratio criteria. His primary objective is to ensure that the robot gets the job done satisfactorily.

Robotic tasks can be described at many levels of abstraction. As an example, consider the task of assembling a Printed Circuit Board (PCB). At the highest level, the task can simply be described in a human-like command as "assemble a PCB". At an intermediate level, the task will be decomposed into

subtasks such as “locate chip A in the part feeder”, “pick up chip A”, and “insert chip A into the board”, etc. At the lowest level, the robot motion sequence and associated trajectory are provided, such as “move the EEF from point A to point B”, “exert a force of k Newton in the X-direction to insert a chip at point C”, etc. For the purpose of this research, we will assume that a low-level task description is given.

Most robotic tasks can be described in terms of numerical task specifications. For example, most non-contact tasks are described by the desired position trajectory (which also implies velocity and acceleration) of the robot’s end-effector. For contact tasks, in addition to the desired position trajectory, the desired force trajectory is also specified. The following sections discuss the task requirements common to many robotic tasks.

4.1.1 Successful Operation

This is the most fundamental of task requirements. Successful operation in this context means that the task is executed without violating any constraint. For example, an operation is considered a failure if any of joint travel limits is exceeded, if the robot collides with an obstacle, or a singularity is approached causing mathematical breakdown during matrix inversion. Constraint violations, at the minimum, will cause the system to fail to complete the desired task. At worse, they could cause damage to the system and/or its surroundings.

4.1.2 EEF Speed

When the desired position trajectory is prescribed, the EEF speed has already been implied. Motion planning is a term describing how to come up with the desired trajectory or path. This takes into account many factors including robot capabilities, work environment, and the task goal. One thing that is not considered is the robot configuration at a given EEF position. Owing to redundancy, the robot

can assume one of an infinite number of configurations given an EEF position. The configuration has a significant impact on the achievable EEF speed. It is imperative that redundancy be resolved in such a manner that the desired overall EEF speed is accommodated.

4.1.3 EEF Force

For a manipulator to be of any use, it needs to do some "work," which usually means that it has to either exert some force to the environment or resist some external force from the tool or payload. Thus the robot's ability to exert force or resist external force is very important. We refer to this as the robot's force capability. The robot's force capability is mainly a function of joints' torque capacities, robot's kinematic and dynamic properties, as well as the robot configuration. As with the EEF speed, redundancy can be used to alter the robot's force capability. In order to perform a successful task, the robot's force capability must exceed the required EEF force.

4.1.4 Accuracy

Accuracy is essential in a variety of robotic tasks including pick and place, cutting, drilling, etc. For free motion, only position accuracy is relevant. For constrained motion, if the environment is modeled as spring-like, then force accuracy is proportional to the position accuracy. Therefore, only position accuracy will be considered here.

Mavroidis *et al.* [1998] suggest that sources of errors in a manipulator are machining errors, assembly errors, deflections, measurement and control, joint errors, clearances. Errors can be categorized as "repeatable" or "random." Repeatable errors are errors whose magnitudes and signs change with robot configuration in a predictable and repeatable manner. Examples of repeatable errors are assembly errors and deflections. Random errors, on the other hand, can

have their magnitudes and signs change unpredictably. Examples of random errors include errors caused by backlash of gear train and encoder resolution. Repeatable errors can in general be compensated for to a certain degree (the goal should be approximately 90%) while random errors cannot easily be dealt with. In this work, we will consider errors from encoder resolution and deflections due to joint flexibility.

Redundancy can be employed to reduce the effects of some of these errors, thus improving the accuracy at the EEF. Two of the simpler usages of redundancy to improve accuracy are (1) to reduce the error propagation from the joint to the EEF and (2) to increase the global stiffness, which would in turn reduce the effect of joint deflections.

4.2 Robot Capabilities

For a robotic task to be performed successfully, we must satisfy all the task requirements, which are defined in terms of desired speed, force, and accuracy at the EEF. It is therefore necessary to be able to estimate the robot's *achievable* speed, force, and accuracy and compare them to the desired values. This section discusses how to estimate these *robot capabilities* from the robot properties, joint capacities, and robot configuration.

4.2.1 Issues Regarding Estimation of Robot Capabilities

This section discusses two primary issues that arise when attempting to develop the formulations for estimating robot capabilities.

4.2.1.1 Ellipsoids and Polytopes

The concept of ellipsoids is perhaps the most popular tool used to analyze the manipulator's manipulability and kinetostatic capability due to its simplicity and ease of interpretation. In the ellipsoid concept, the set of realizable EEF

velocities from the set of joint velocities whose Euclidean norm is less than or equal to one can be obtained by transforming the space $Q_e = \left\{ \dot{\theta} : \left\| \dot{\theta} \right\|_2 \le 1 \right\}$ via

$$\dot{x} = J\dot{\theta} \,. \tag{4.1}$$

Note that this transformation changes the Euclidean norm which is a unit hypersphere in the joint space to an ellipsoid in the task space. Figure 4-1 illustrates the ellipsoid concept for a 2-DOF planar manipulator with the task space of dimension 2. $\sigma_{\min}$ and $\sigma_{\max}$ are, respectively, the minimum and maximum singular values of the Jacobian, which correspond to the directions of minimum and maximum task space velocities, respectively.

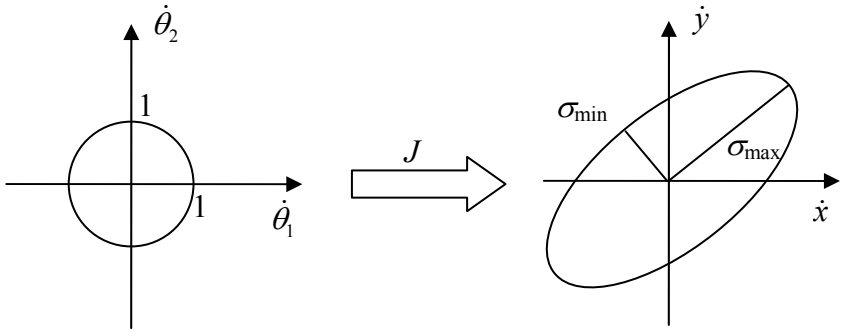


Figure 4-1: A 2-DOF example of ellipsoid transformation

The ellipsoid transformation can be used to estimate the robot EEF capabilities given the joint (or actuator) capabilities as will be shown shortly. However, it is noteworthy that an ellipsoid only approximates a more accurate *polytope*. A polytope is the result of transformation from a hypercube via a linear transformation in the same way an ellipsoid is from a hypersphere. Since actuators have maximum/minimum performance values (speed limits, torque limits, etc.), a more accurate representation of their capabilities is by the ∞-norm (or a hypercube) not the Euclidean norm (or a hypersphere). Ellipsoids can be considered *conservative* approximations of polytopes because they always yield smaller EEF capabilities. Figure 4-2 compares the ellipsoid and the polytope of a 2-DOF planar

robot. In addition, Lee [1997] also showed that the directions of minimum and maximum velocity transmission ratios obtained via the ellipsoid method may not be the same as those obtained from the polytope method.

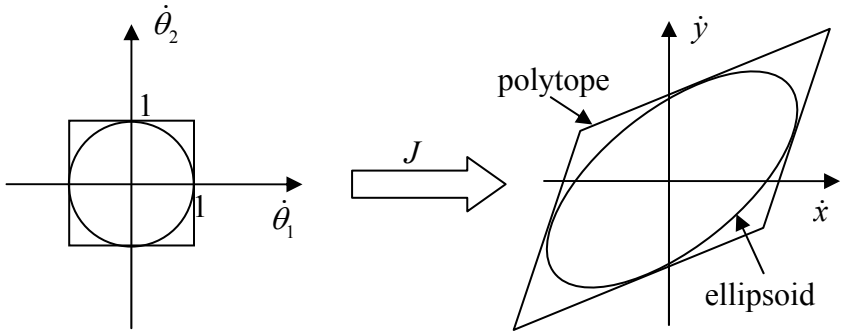


Figure 4-2: A 2-DOF example of ellipsoid and polytope

Although polytopes are a more accurate representation than ellipsoids, they are also complex, not analytically tractable, and considerably more expensive to compute, especially for redundant manipulators. Using a new recursive dimension-growing algorithm, Hwang *et al*. [2000] showed that the computation time of a velocity polytope of a 10-DOF robot on to a 3D task space was improved from 427.52 seconds using a conventional method to 6.64 seconds on an HP 700 system. Even with the remarkable improvement in computation speed, it is still far from being acceptable for real-time applications. Many researchers including Chiacchio *et al.* [1997], Lee [1997], and Finotello *et al.* [1998] have applied the concept of polytopes to analyzing kinetostatic capabilities of manipulators.

In their investigation of the acceleration characteristics of non-redundant manipulators, Bowling and Khatib [1995] used the *ellipsoid expansion* approach. In this approach, "the linear and angular accelerations are considered as two separate entities." The decomposition of the task space will be discussed in detail in the next section. A unique trait of this approach, however, is that quantities of interest are mapped from the task space to the joint space whereas other studies do the opposite.

Figure 4-3 illustrates how the ellipsoid expansion method works. The purpose here is to find the largest possible magnitude of acceleration in all directions that the manipulator can provide. This isotropic acceleration is represented by a hypersphere with radius *a* as shown in the right hand side of Figure 4-3. This hypersphere is then mapped from the task space to the joint space as an ellipsoid as shown in the left hand side of Figure 4-3. The isotropic acceleration magnitude *a* is determined by expanding/contracting the ellipsoid until it lies within and is tangent to one or more of the torque bounds.

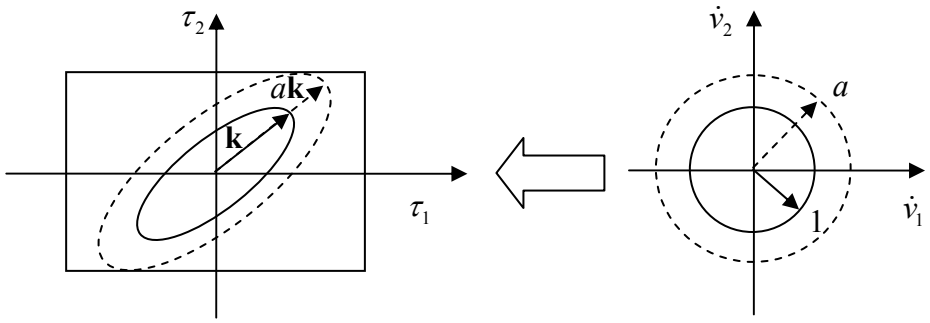


Figure 4-3: Ellipsoid expansion and reverse mapping (Adapted from Bowling and Khatib [1995]).

With this reverse mapping, the computationally expensive process of creating polytopes in the task space is eliminated. Yet, bounds on joint capabilities are represented and used in its simplest and exact form as a hypercube. Thus, the ellipsoid expansion method has a benefit of being more accurate than the ellipsoid model without the computational overhead of the polytope. Moreover, an important piece of information obtained via this method that is not possible in the ellipsoid method is the limiting joint, i.e. the joint whose capability limits the EEF capability. In Figure 4-3 above, Joint 2 is the limiting joint because the expanded ellipsoid touches the torque bound of Joint 2. The limiting joint information is extremely valuable in the design phase of the robot. However, this approach does not directly provide the visualization of the robot capabilities in the task space that ellipsoids or polytopes can. Perhaps more importantly, it fails to impart the general

characteristics of the robot capabilities in the task space such as the directions of maximum or minimum transmission ratios, etc. Nonetheless, for our purpose of estimating the robot capabilities in a given direction, the ellipsoid expansion approach may be more appropriate than either the ellipsoid or the polytope method.

4.2.1.2 Homogeneity of the Jacobian Matrix

Another important aspect that has to be pointed out here is that the measures whose values depend upon the manipulator Jacobian J, JJ^T, or the pseudoinverse of J suffer from possible inconsistency deriving from improper use of vector norm and from dependency on change of scale and coordinate frame [Doty *et al.*, 1995]. Chiacchio and Concilio [1998] avoided this problem when formulating a dynamic manipulability ellipsoid by assuming that all the joints are of the same kind (prismatic or revolute) and that the task space is composed by either linear or angular motion. Yoshikawa [1991] decomposed the task space and the Jacobian matrix into the translation and rotation parts and proposed the translational and rotational manipulability measures. Combining this Jacobian decomposition with normalizing the input vector gives us *homogeneous* Jacobian matrices (i.e. Jacobian matrices whose elements are of the same units). Consider the following example. Suppose that the robot has two joints with the first joint being revolute and the second prismatic. Furthermore, the task space has two outputs; the first is translation and the other rotation. The dimension analysis of Eq. (4.1) yields[10]

$$\begin{bmatrix} length/time \\ 1/time \end{bmatrix} = \begin{bmatrix} length & 1 \\ 1 & 1/length \end{bmatrix} \begin{bmatrix} 1/time \\ length/time \end{bmatrix}. \tag{4.2}$$

[10] Note that prismatic joints do not contribute to rotational motion and thus the corresponding Jacobian elements should be zero. However, for consistency in the dimension analysis, their dimensions are shown here.

Obviously, the Jacobian contains mixed units. Now, let's decompose the Jacobian into the translation and rotation parts such that Eq. (4.1) becomes

$$\begin{aligned} \dot{x}_T &= J_T \dot{\theta} \\ \dot{x}_R &= J_R \dot{\theta} \end{aligned}. \tag{4.3}$$

where $\dot{x}_T$ and $\dot{x}_R$ are, respectively, the m_T-dimension translational velocity and the m_R-dimension rotational velocity of the end-effector; J_T and J_R are the translational and rotational Jacobian matrices, respectively. Now the dimension analysis yields

$$\begin{aligned} length/time &= \begin{bmatrix} length & 1 \end{bmatrix} \begin{bmatrix} 1/time \\ length/time \end{bmatrix} \\ 1/time &= \begin{bmatrix} 1 & 1/length \end{bmatrix} \begin{bmatrix} 1/time \\ length/time \end{bmatrix} \end{aligned}. \tag{4.4}$$

These decomposed Jacobian matrices (J_T and J_R) still contain mixed units. The next step is to normalize the input vector (joint velocities) by their speed limits. Let's define the normalized joint velocity vector as

$$\tilde{\dot{\theta}} = L_{\dot{\theta}}^{-1} \dot{\theta}, \tag{4.5}$$

where $L_{\dot{\theta}} = diag\left\{\dot{\theta}_1^{\max}, \dot{\theta}_2^{\max}, \ldots, \dot{\theta}_n^{\max}\right\}$ is an $n \times n$ diagonal matrix consisting of the joint speed limits. Note that the normalized input vector $\tilde{\dot{\theta}}$ is dimensionless. This normalization process is not only necessary for a mixed-joint robot but also useful for a same-joint (either revolute or prismatic) robot with each joint possessing a different capability limit. It is assumed here that the joint speed upper and lower limits are of equal magnitude but in the opposite direction. Substituting $\dot{\theta}$ from Eq. (4.5) into Eq. (4.3) yields

$$\begin{aligned} \dot{x}_T &= J_T L_{\dot{\theta}} \tilde{\dot{\theta}} \triangleq \tilde{J}_T \tilde{\dot{\theta}} \\ \dot{x}_R &= J_R L_{\dot{\theta}} \tilde{\dot{\theta}} \triangleq \tilde{J}_R \tilde{\dot{\theta}} \end{aligned} \tag{4.6}$$

The new dimension analysis yields

$$
\begin{aligned}
length/time &= \begin{bmatrix} length & 1 \end{bmatrix} \begin{bmatrix} 1/time & 0 \\ 0 & length/time \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} length/time & length/time \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
1/time &= \begin{bmatrix} 1 & 1/length \end{bmatrix} \begin{bmatrix} 1/time & 0 \\ 0 & length/time \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 1/time & 1/time \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\end{aligned}
\tag{4.7}
$$

The new Jacobian matrices ($\tilde{J}_T$ and $\tilde{J}_R$) now have consistent units and therefore any measures derived from these matrices do not suffer the same potential inconsistency as those derived directly from the manipulator Jacobian *J*.

In the development of the estimation of robot capabilities, polytopes will not be utilized due to their unacceptable computation time as pointed out earlier. We will opt instead for the ellipsoid model and the vector expansion method, which is adapted from the ellipsoid expansion method proposed by Bowling and Khatib [1995]. As its name implies, this method expands or contracts a vector, instead of an ellipsoid, to determine the robot capabilities. Like the ellipsoid expansion method, the vector expansion method has a benefit of being more accurate than the ellipsoid model without the computational overhead of the polytope. Also, the limiting joint information is still available as it is in the ellipsoid expansion method. The normalized decomposed Jacobian matrices will be used whenever appropriate to promote the consistency of the results.

In our development, for each capability, we will discuss the formulations and give numerical examples. These examples are calculated for the Planar4R robot. The examples in general include the robot capability values at a fixed configuration and the best and worst robot capability values for a given EEF position. The fixed configuration at $\theta = \begin{bmatrix} 45 & -45 & -45 & -45 \end{bmatrix}^T$ shown in the figure below will be used in all of the examples.

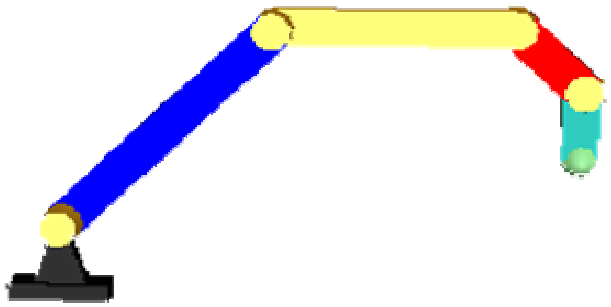


Figure 4-4: Planar4R robot at the fixed configuration.

For a given EEF position, only X and Y locations are fixed. The EEF orientation is not. The best and worst robot capability values are determined by varying the first joint from one extreme to the other while keeping the EEF stationary. Obviously, these may not be the best and worst possible robot capability values since the whole null space is not searched. However, this still gives us an idea of how big an impact that the robot configuration has on the robot capabilities and how we can exploit redundancy to enhance them.

4.2.2 Achievable EEF Speed

Based upon the joint speed limits, this section presents two methods that can be used to estimate the EEF speed in the task space from a purely kinematic point of view.

4.2.2.1 Ellipsoid Formulation

This section formulates the EEF achievable speed estimation using the ellipsoid concept. Based upon the decomposed Jacobian matrices in Eq. (4.3), Yoshikawa [1991] defined the Translational Velocity Ellipsoid (TVE)[11] *in the*

[11] In Yoshikawa [1991], the term Translational Manipulability Ellipsoid (TME) was used but we think the term Translational Velocity Ellipsoid is more appropriate and thus will be used here.

weak sense as "the set of all translational velocities that are realizable under the constraint $\left\|\dot{\theta}\right\|_2 \le 1$." The TVE *in the strong sense* adds another constraint that the EEF orientation is kept constant ($\dot{x}_R = 0$). The Rotational Velocity Ellipsoids (RVEs) in the weak and strong senses are defined similarly. However, as we have shown previously, the decomposed Jacobian matrices in Eq. (4.3) are still not homogeneous. It is better to use the normalized decomposed Jacobian matrices in Eq. (4.6).

Because the analyses are identical, we will only develop the formulations on the translational EEF velocity and will deduce the results for the rotational EEF velocity. The translation part of Eq. (4.6) is repeated here for convenience.

$$\dot{x}_T = J_T L_{\dot{\theta}} \tilde{\dot{\theta}} \triangleq \tilde{J}_T \tilde{\dot{\theta}} \tag{4.8}$$

Using the pseudoinverse, we obtain

$$\tilde{\dot{\theta}} = \tilde{J}_T^{\dagger} \dot{x}_T \,. \tag{4.9}$$

Then,

$$\left\|\tilde{\dot{\theta}}\right\|_2 = \tilde{\dot{\theta}}^T \tilde{\dot{\theta}} = \dot{x}_T^T \left(\tilde{J}_T^{\dagger}\right)^T \tilde{J}_T^{\dagger} \dot{x}_T \,. \tag{4.10}$$

Therefore, the TVE in the weak sense is described by

$$\left\{\dot{x}_T : \dot{x}_T^T \left(\tilde{J}_T^{\dagger}\right)^T \tilde{J}_T^{\dagger} \dot{x}_T \le 1 \text{ and } \dot{x}_T \in R\left(\tilde{J}_T\right)\right\} \tag{4.11}$$

where $R\left(\tilde{J}_T\right)$ denotes the range of $\tilde{J}_T$. If the manipulator is not in a singular configuration, then the condition $\dot{x}_T \in R\left(\tilde{J}_T\right)$ is not necessary. Rearranging a few terms and Eq. (4.11) can be rewritten as

$$\left\{\dot{x}_T : \dot{x}_T^T \left(J_T L_{\dot{\theta}}^2 J_T^T\right)^{-1} \dot{x}_T \le 1 \text{ and } \dot{x}_T \in R\left(\tilde{J}_T\right)\right\}. \tag{4.12}$$

Let $\dot{x}_T = v\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then, the inequality in Eq. (4.12) becomes

$$\left[\hat{t}^T \left(J_T L_{\dot{\theta}}^2 J_T^T\right)^{-1} \hat{t}\right] v^2 \le 1. \tag{4.13}$$

Thus, the maximum achievable translational EEF speed in the $\hat{t}$ direction can be estimated by

$$v_{\max} = \pm \frac{1}{\sqrt{\hat{t}^T \left(J_T L_{\dot{\theta}}^2 J_T^T\right)^{-1} \hat{t}}}. \tag{4.14}$$

Similarly, the maximum achievable rotational EEF speed in the $\hat{t}$ direction is

$$\omega_{\max} = \pm \frac{1}{\sqrt{\hat{t}^T \left(J_R L_{\dot{\theta}}^2 J_R^T\right)^{-1} \hat{t}}}. \tag{4.15}$$

4.2.2.2 Vector Expansion Formulation

This section discusses how to estimate the manipulator's speed capabilities using the *vector expansion* method. Note from Eq. (4.8) that the bounds on $\tilde{\dot{\theta}}$ can be written as

$$-\mathbf{1} \le \tilde{\dot{\theta}} \le \mathbf{1} \tag{4.16}$$

where **1** is a vector of *n*-dimension with each element equal to one. Combining Eqs. (4.9) and (4.16) gives

$$-\mathbf{1} \le \tilde{J}_T^{\dagger} \dot{x}_T \le \mathbf{1}. \tag{4.17}$$

Again, let $\dot{x}_T = v\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. The maximum achievable speed is conceptually determined by expanding/contracting the vector $v\tilde{J}_T^{\dagger}\hat{t}$ (changing *v*) until it touches one of the joint speed limits as depicted in Figure 4-5.

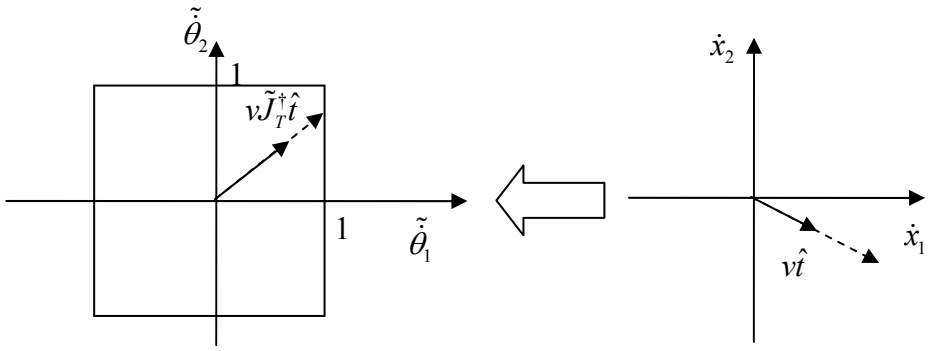


Figure 4-5: Joint speed limits and velocity vector expansion.

In determining $v_{\max}$ (by expanding or contracting $v\tilde{J}_T^{\dagger}\hat{t}$), we can compare; the vectors $\tilde{J}_T^{\dagger}\hat{t}$ and **1** element by element. By inspecting Eq. (4.17), the maximum achievable translational EEF speed in the $\hat{t}$ direction can be expressed as

$$v_{\max} = \min_i \frac{1}{\left|\tilde{J}_T^{\dagger}\hat{t}\right|_i}, \quad i = 1,\ldots,n \tag{4.18}$$

where $\left|\tilde{J}_T^{\dagger}\hat{t}\right|_i$ is the absolute value of the ith-component of the vector $\tilde{J}_T^{\dagger}\hat{t}$. Similarly, the maximum achievable rotational EEF speed in the $\hat{t}$ direction is

$$\omega_{\max} = \min_i \frac{1}{\left|\tilde{J}_R^{\dagger}\hat{t}\right|_i}, \quad i = 1,\ldots,n. \tag{4.19}$$

4.2.2.3 Example

Assume that the joint speed limits are given in the matrix form as $L_{\dot{\theta}} = diag\{50,50,50,50\}$ (all joint speed limits are 50 deg/s). Figure 4-6 shows the velocity ellipsoid (created by the ellipsoid method) and the velocity polytope (created by the vector expansion method) of Planar4R robot at the configuration $\theta = \begin{bmatrix} 45 & -45 & -45 & -45 \end{bmatrix}^T$. The numbers beside the edges of the polytope indicate the limiting joints at those edges. Although the vector expansion method

cannot directly create a polytope, the polytope here was generated by incrementing the angle of the unit vector $\hat{t}$ by one degree at a time and recording the resulting robot's speed capability. The ellipsoid was generated the same way even though we could have easily used Eq. (4.13) to plot the ellipsoid. As expected, the ellipsoid is smaller than the polytope. The ellipsoid gives a fairly good estimation in the Y direction (0.5260 m/s vs 0.5756 m/s) whereas the estimation in the X direction is far too small (0.2026 m/s vs 0.3222). Interestingly, Joint 4 is never the limiting joint in any direction at this configuration.

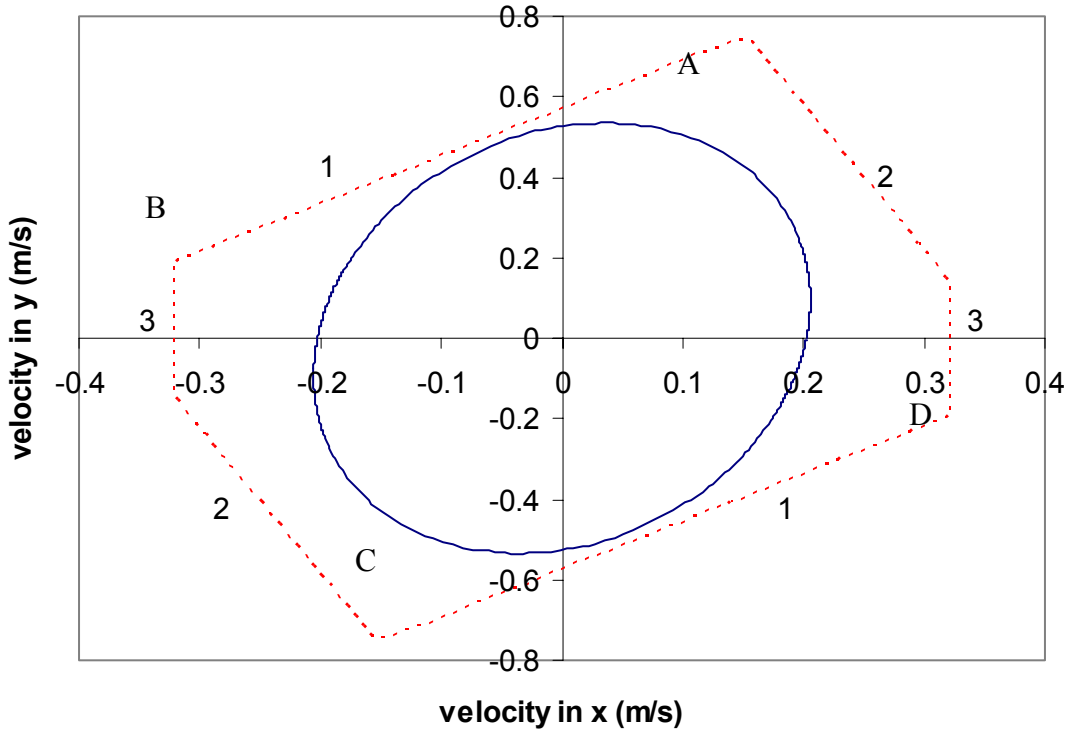


Figure 4-6: EEF velocity ellipsoid (solid) and polytope (dotted).

Now, let's assume that Joint 1 suddenly fails and needs to be locked ($\dot{\theta}_1^{\max} = 0$). The joint speed limit matrix now becomes $L_{\dot{\theta}} = diag\{0, 50, 50, 50\}$. We wish to determine how this would affect the robot speed capabilities and if the proposed method is flexible enough to cope with this unforeseen situation. Before

running this test, let's investigate what kind of result we would anticipate with Joint 1 being disabled. First of all, we would certainly expect a diminished speed capability. But in what direction and by how much? A look at the ellipsoid in Figure 4-6 does not give any insight to these questions. Fortunately, the polytope generated by the vector expansion method provides some insight. As can be seen from the polytope in Figure 4-6, Joint 1 is already the limiting joint for approximately half of the polytope (from points A to B and C to D). Reducing or eliminating its capability should diminish the robot's speed capability significantly.

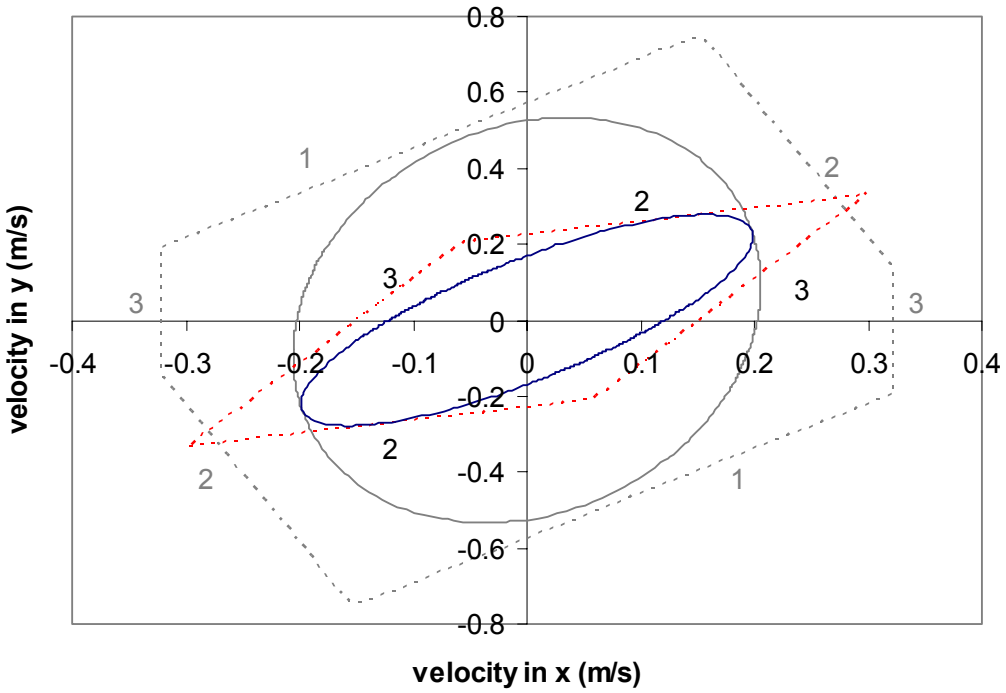


Figure 4-7: Velocity ellipsoid (solid) and polytope (dotted) with Joint 1 disabled.

The EEF speed capability of the Planar4R robot with Joint 1 disabled is shown in Figure 4-7. Also portrayed in the figure in a light color are the original ellipsoid and polytope for comparison purposes. As expected, both the ellipsoid

and polytope shrink considerably. This demonstrates the applicability of our proposed method to fault tolerance and failure recovery.

If the EEF position is fixed at the point (0.5, 0.0), then using the vector expansion method the minimum and maximum achievable EEF speeds in the X direction are found to be 0.1705 m/s and 0.3710 m/s, respectively and in the Y direction are 0.4363 m/s and 0.5529 m/s. The configurations corresponding to these speeds are shown in Figure 4-8. Apparently, at this particular EEF location, the robot configuration has a huge impact on the speed capability in the X direction but not as much in the Y direction.

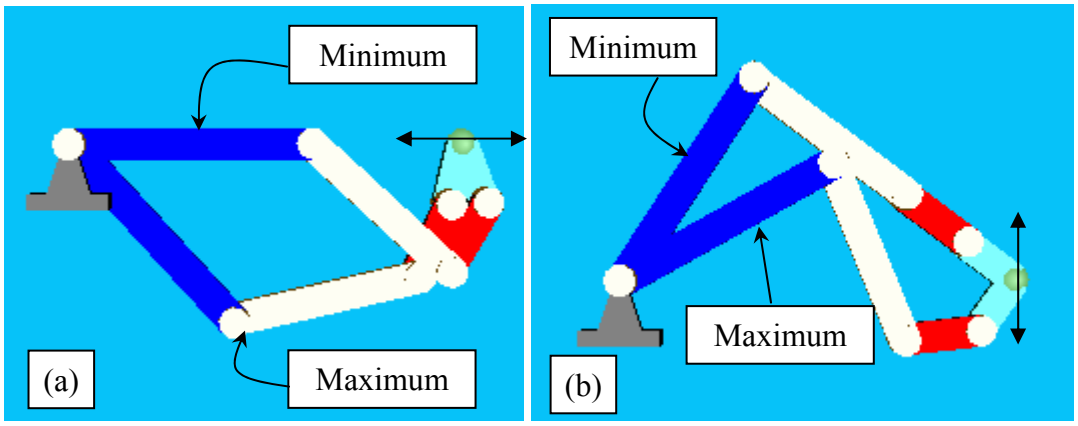


Figure 4-8: The configurations with maximum and minimum achievable EEF speeds in (a) X direction and (b) Y direction.

4.2.3 Achievable EEF Acceleration

The previous section discussed the achievable EEF speed from a purely kinematic point of view. The true achievable EEF speed also depends on how much the robot can accelerate its EEF. In this section, we wish to determine the achievable EEF acceleration, which is a function of the robot's joint torque limits and its dynamic properties.

4.2.3.1 Ellipsoid Formulation

The formulations here are adapted from the work done on dynamic transmissibility by Chiacchio and Concilio [1998]. The second-order differential kinematics of a manipulator can be written as

$$\ddot{x} = J\ddot{\theta} + \dot{J}\dot{\theta} \tag{4.20}$$

The robot dynamics equation can be written as

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + g(\theta) + J^T F_e = \tau \tag{4.21}$$

where $\tau \in \mathcal{R}^n$ is a generalized joint torque vector, $M(\theta) \in \mathcal{R}^{n\times n}$ a symmetric and positive definite joint inertia matrix, $C(\theta,\dot{\theta})\dot{\theta} \in \mathcal{R}^n$ the Coriolis/centripetal torque, $g(\theta) \in \mathcal{R}^n$ the torque due to gravity, and $F_e \in \mathcal{R}^m$ the contact force exerted by the EEF.

To study the robot's acceleration capability, we assume that the robot is stationary ($\dot{\theta} = 0$) and the EEF is not constrained (F_e = 0). Thus, Eq.(4.20) becomes

$$\ddot{x} = J\ddot{\theta}, \tag{4.22}$$

and Eq. (2.27) is reduced to

$$M(\theta)\ddot{\theta} + g(\theta) = \tau. \tag{4.23}$$

According to Khatib [1990],

Define the normalized joint torque vector

$$\tilde{\tau} = L_\tau^{-1}\tau \tag{4.24}$$

where $L_\tau = diag\left\{\tau_1^{\max}, \tau_2^{\max}, \ldots, \tau_n^{\max}\right\}$ is a $n \times n$ diagonal matrix consisting of the joint torque limits. Combining Eqs. (4.22)-(4.24) yields

$$\ddot{x}_T = B_T\tilde{\tau} + \ddot{x}_{gT} \tag{4.25}$$

where $B_T = J_T M^{-1} L_\tau$ and $\ddot{x}_{gT} = -J_T M^{-1} g$. Using the pseudoinverse of B_T in Eq. (4.25),

$$\tilde{\tau} = B_T^{\dagger}\left(\ddot{x}_T - \ddot{x}_{gT}\right) \tag{4.26}$$

$\tilde{\tau}$ in Eq. (4.26) can be thought of as a set of joint torques required to generate the translational (linear) acceleration $\ddot{x}_T$ in the task space. Therefore, it is technically more correct to denote it as $\tilde{\tau}_T$. Similarly, the joint torques required to generate the rotational (angular) acceleration $\ddot{x}_R$ can be expressed as

$$\tilde{\tau}_R = B_R^{\dagger}\left(\ddot{x}_R - \ddot{x}_{gR}\right) \tag{4.27}$$

where $B_R = J_R M^{-1} L_\tau$ and $\ddot{x}_{gR} = -J_R M^{-1} g$. The total normalized torque is then the sum of the two $\tilde{\tau} = \tilde{\tau}_T + \tilde{\tau}_R$.

The unit sphere in the space of normalized joint torques $\tilde{\tau}^T \tilde{\tau} \le 1$ can now be mapped to the task space to give the translational acceleration ellipsoid as

$$\left(\ddot{x}_T + J_T M^{-1} g\right)^T B_T^{\dagger T} B_T^{\dagger}\left(\ddot{x}_T + J_T M^{-1} g\right) \le 1. \tag{4.28}$$

Defining $Q = ML_\tau^{-2} M$, Eq. (4.28) can be rewritten as

$$\left(\ddot{x}_T + J_T M^{-1} g\right)^T \left(J_T Q^{-1} J_T^T\right)^{-1}\left(\ddot{x}_T + J_T M^{-1} g\right) \le 1. \tag{4.29}$$

Let $N_T \equiv \left(J_T Q^{-1} J_T^T\right)^{-1}$ and $\ddot{x}_T = a_T \hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then Eq. (4.29) can be rewritten as

$$\alpha_T f^2 + 2\beta_T f + \gamma_T \le 0 \tag{4.30}$$

where

$$\begin{aligned} \alpha_T &= \hat{t}^T N_T \hat{t} \\ \beta_T &= -\hat{t}^T N_T \ddot{x}_{gT} \\ \gamma_T &= \ddot{x}_{gT}^T N_T \ddot{x}_{gT} - 1 \end{aligned} .$$

Solving Eq. (4.30) yields the estimation of the achievable translational EEF acceleration given by

$$\frac{-\beta_T - \sqrt{\beta_T^2 - \alpha_T \gamma_T}}{\alpha_T} \le a_T \le \frac{-\beta_T + \sqrt{\beta_T^2 - \alpha_T \gamma_T}}{\alpha_T}. \tag{4.31}$$

The rotational EEF acceleration capability can be estimated using Eqs. (4.29)-(4.31) with J_T being replaced with J_R.

4.2.3.2 Vector Expansion Method

The bounds on the normalized joint torques can be written as

$$-\mathbf{1} \le \tilde{\tau} \le \mathbf{1} \tag{4.32}$$

where **1** is a vector of n-dimension with each element equal to one. Substituting $\tilde{\tau}$ from Eq. (4.26) in Eq. (4.32) yields

$$-\mathbf{1} \le B_T^{\dagger}\left(\ddot{x}_T - \ddot{x}_{gT}\right) \le \mathbf{1}. \tag{4.33}$$

Rearranging a few terms gives the governing equation as

$$\tau_{lower} \le B_T^{\dagger}\ddot{x}_T \le \tau_{upper} \tag{4.34}$$

where $\tau_{upper} = \mathbf{1} + B_T^{\dagger}\ddot{x}_{gT}$ and $\tau_{lower} = -\mathbf{1} + B_T^{\dagger}\ddot{x}_{gT}$.

Let $\ddot{x}_T = a\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then, Eq. (4.34) can be rewritten as

$$\tau_{lower} \le aB_T^{\dagger}\hat{t} \le \tau_{upper} \tag{4.35}$$

The vector expansion model describing Eq. (4.35) is illustrated in Figure 4-9.

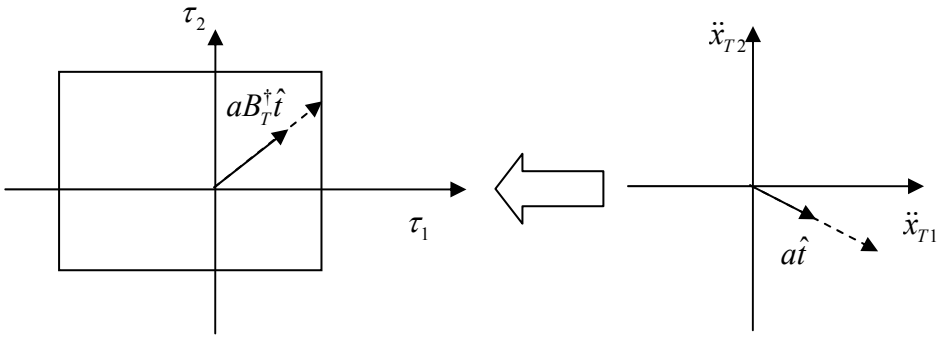


Figure 4-9: Joint torque bounds and acceleration vector expansion.

The torque bounding box is shifted due to gravity. Note here that, unlike the joint speed bounds, the torque bounds in Figure 4-9 are not symmetrical due to

the weight of the robot. If the bounding box is shifted so far that it does not contain the origin, it means the robot cannot support its own weight. This should never happen in real operations. Mathematically, this means that all elements of τ_{lower} have to remain negative and τ_{upper} positive. But this situation could arise if we use this analysis as a tool in designing robots. By expanding/contracting the vector $aB_T^{\dagger}\hat{t}$, the magnitude of the achievable translational acceleration is determined by

$$a \le \min_i \left\{ \max \left(\frac{\tau_{upper,i}}{\left[B_T^{\dagger}\hat{t} \right]_i}, \frac{\tau_{lower,i}}{\left[B_T^{\dagger}\hat{t} \right]_i} \right) \right\}, \quad i = 1, \ldots, n \tag{4.36}$$

where $\tau_{lower,i}$, $\tau_{upper,i}$, and $\left[B_T^{\dagger}\hat{t} \right]_i$ are the i^{th} elements of vectors τ_{lower}, τ_{upper}, and $B_T^{\dagger}\hat{t}$, respectively. For each joint *i*, note that one of $\frac{\tau_{upper,i}}{\left[B_T^{\dagger}\hat{t} \right]_i}$ and $\frac{\tau_{lower,i}}{\left[B_T^{\dagger}\hat{t} \right]_i}$ will be positive and the other negative. The max operator will choose the positive one between them, which will present the maximum acceleration allowed by joint *i*. Then, the min operator will choose the maximum acceleration achievable by the whole robot.

Similarly, the rotational EEF acceleration capability can be expressed as

$$\alpha \le \min_i \left\{ \max \left(\frac{\tau_{upper,i}}{\left[B_R^{\dagger}\hat{t} \right]_i}, \frac{\tau_{lower,i}}{\left[B_R^{\dagger}\hat{t} \right]_i} \right) \right\}, \quad i = 1, \ldots, n . \tag{4.37}$$

4.2.3.3 Example

We wish to determine the EEF acceleration capability of the Planar4R robot at the configuration $\theta = \begin{bmatrix} 45 & -45 & -45 & -45 \end{bmatrix}^T$. Let the joint torque limit matrix $L_\tau = diag\{100, 45, 35, 15\}$ N-m. Figure 4-10 shows the acceleration ellipsoid and polytope of the Planar4R robot. Not surprisingly, both the ellipsoid and polytope are shifted downwards because of gravity (in fact, they are also slightly shifted to

the right). Again, the numbers beside the edges of the polytope denote the limiting joints. For this particular case, only Joints 2 and 4 are limiting joints.

If the EEF position is fixed at the point (0.5, 0.0), then using the vector expansion method the minimum and maximum achievable EEF accelerations in the X direction are found to be 21.40 m/s^2 and 95.72 m/s^2, respectively and in the Y direction are 11.49 m/s^2 and 56.29 m/s^2. The configurations corresponding to these accelerations are shown in Figure 4-11. Apparently, at this particular EEF location, the robot configuration has a huge impact on the acceleration capability in both X and Y directions.

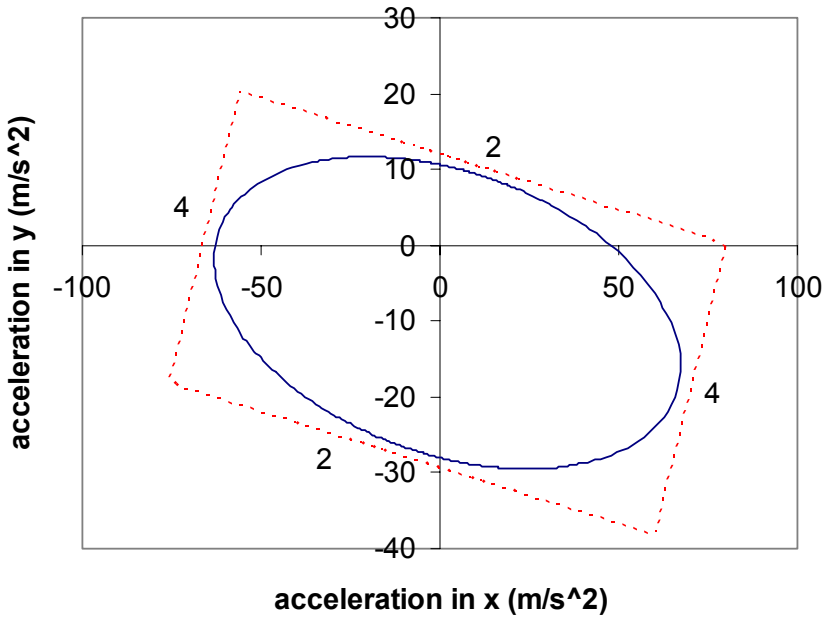


Figure 4-10: Acceleration ellipsoid (solid) and polytope (dotted).

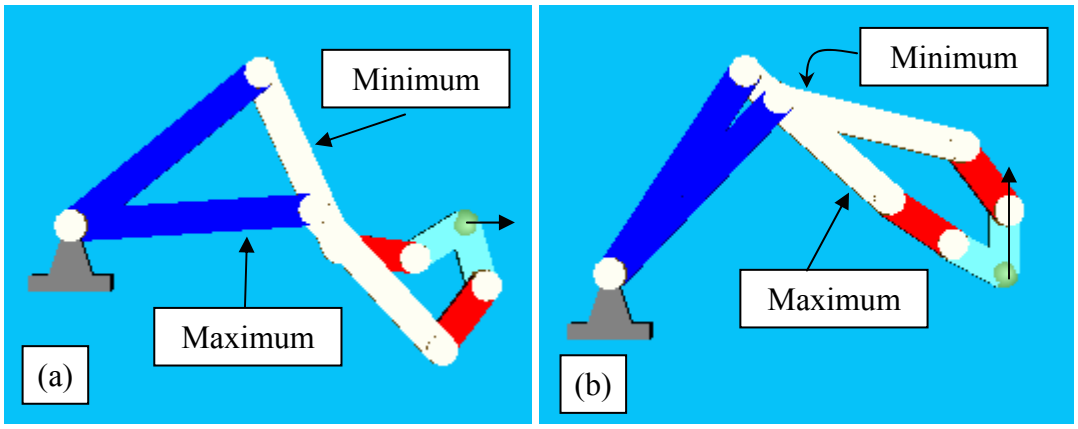


Figure 4-11: The configurations with maximum and minimum achievable EEF accelerations in (a) X direction and (b) Y direction

4.2.4 Maximum EEF Position Error

If the joint errors $\Delta\theta$ are assumed to be small, the EEF error can then be estimated by

$$\Delta x = J\Delta\theta\,. \tag{4.38}$$

This is the same equation that governs the relationship between the joint and EEF velocities. Thus, the process of estimating the maximum EEF position error is identical to that of estimating the achievable EEF speed.

4.2.4.1 Ellipsoid Formulation

Similar to the EEF speed, the translational EEF error Δx_T is constrained by

$$\Delta x_T^T \left(J_T L_{\Delta\theta}^2 J_T^T\right)^{-1} \Delta x_T \le 1\,. \tag{4.39}$$

where $L_{\Delta\theta} = diag\left\{\Delta\theta_1^{\max}, \Delta\theta_2^{\max}, \ldots, \Delta\theta_n^{\max}\right\}$ is a $n \times n$ diagonal matrix consisting of the maximum joint errors. Similar to the achievable EEF speed in the previous section, the maximum translational and rotational EEF position errors can then be estimated as

$$\left|\Delta x_T\right|_{\max} = \frac{1}{\sqrt{\hat{t}^T \left(J_T L_{\Delta\theta}^2 J_T^T\right)^{-1} \hat{t}}}$$
$$\left|\Delta x_R\right|_{\max} = \frac{1}{\sqrt{\hat{t}^T \left(J_R L_{\Delta\theta}^2 J_R^T\right)^{-1} \hat{t}}} \quad . \tag{4.40}$$

4.2.4.2 Vector Expansion Formulation

Using the results from Eqs. (4.18) and (4.19), the maximum translational and rotational EEF position errors using the vector expansion method can then be estimated as

$$\left|\Delta x_T\right|_{\max} = \min_i \frac{1}{\left|\tilde{J}_T^{\dagger}\hat{t}\right|_i}, \quad i = 1,\ldots,n$$
$$\left|\Delta x_R\right|_{\max} = \min_i \frac{1}{\left|\tilde{J}_R^{\dagger}\hat{t}\right|_i}, \quad i = 1,\ldots,n \tag{4.41}$$

Although there are many more types of errors, in this work, we assume that each maximum joint error can be estimated by combining the joint encoder resolution with the joint deflection due to joint flexibility.

$$\Delta\theta_i^{\max} = \varepsilon_i + \left[C\tau\right]_i \tag{4.42}$$

where ε_i is the encoder resolution of joint i, C is the joint compliance matrix, and τ is the joint torque vector. In the static case, the joint torques can be obtained from the gravity torques and the external EEF force. Naturally, others joint errors can be easily added to this model.

4.2.4.3 Example

Again, consider the same robot at the same configuration as above. Assume that the encoders of all the joints have a resolution of 0.003 deg, that all joints are flexible and have stiffness of 10^{-5} rad/N-m, that the gravity is in the vertical (or -Y) direction, and that there is no external force. Figure 4-12 shows the error ellipsoid

and polytope for the Planar4R robot. Note that the EEF position error in the Y direction is about 5 times (ellipsoid) to 7 times (polytope) as large as that in the X direction due mostly to the weight of the robot itself.

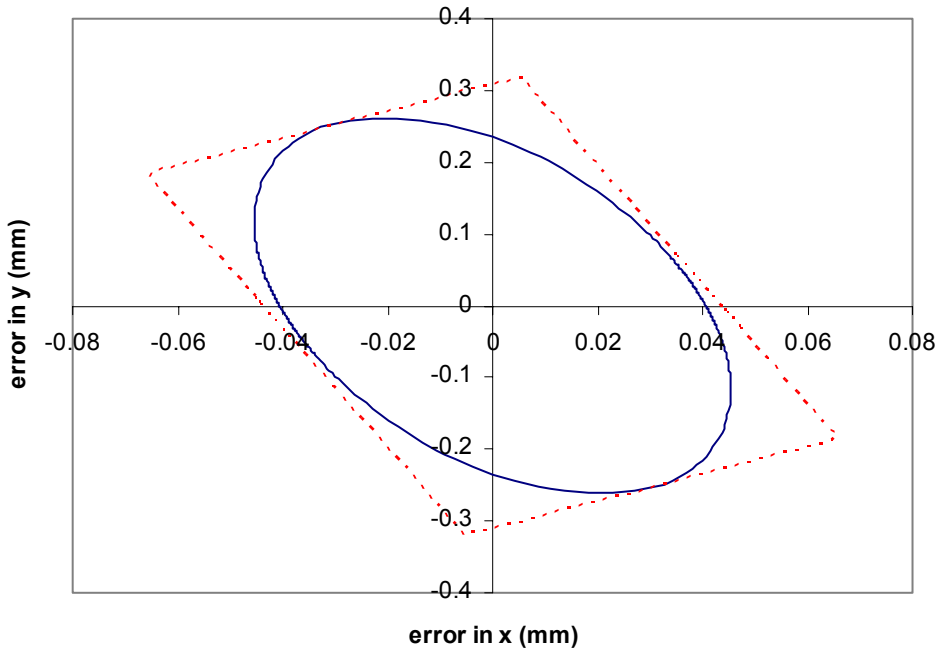


Figure 4-12: EEF error ellipsoid (solid) and polytope (dotted).

Now if the EEF position is fixed at the point (0.5, 0.0), then the minimum and maximum EEF position errors in the X direction are 0.0117 mm and 0.0522 mm, respectively and in the Y direction are 0.199 mm and 0.284 mm. The corresponding configurations are shown in Figure 4-13.

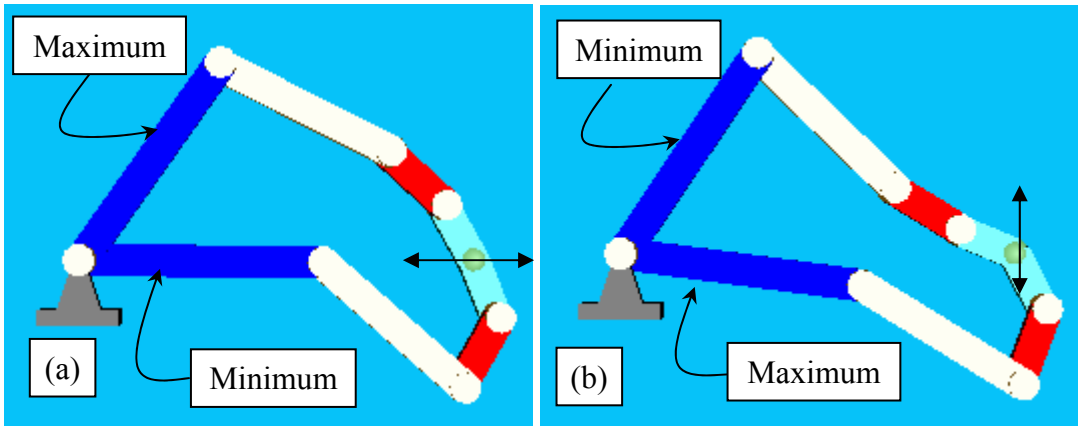


Figure 4-13: The configurations with maximum and minimum EEF position errors in (a) X direction and (b) Y direction.

4.2.5 Achievable EEF Static Force Capability

Many studies found in the literature omit the effect of gravity when computing force capability (see Chiacchio *et al*. [1997], Finotello *et al*. [1998], and Kim and Choi [1999] for example). As Hernandez and Tesar [1996] pointed out that the weight of the manipulator is usually the most dominant load, especially for industrial robots, it is not wise to ignore the gravity effect when estimating the robot force capability. The impact of the gravity on the force capability will be demonstrated shortly.

4.2.5.1 Ellipsoid Formulation

Considering the EEF force and gravity, the joint torques can be expressed as

$$\tau = J^T F + g \tag{4.43}$$

Decomposing the Jacobian and the EEF force yields

$$\tau = \begin{bmatrix} J_T^T & J_R^T \end{bmatrix} \begin{bmatrix} F_T \\ F_R \end{bmatrix} + g = J_T^T F_T + J_R^T F_R + g \,. \tag{4.44}$$

Define the normalized joint torque vector

$$\tilde{\tau} = L_\tau^{-1} \tau \tag{4.45}$$

where $L_\tau = diag\left\{\tau_1^{\max}, \tau_2^{\max}, \ldots, \tau_n^{\max}\right\}$ is a $n \times n$ diagonal matrix consisting of the joint torque limits. The force ellipsoid is then described by

$$\tilde{\tau}^T \tilde{\tau} = \left(J_T^T F_T + J_R^T F_R + g\right)^T L_\tau^{-2} \left(J_T^T F_T + J_R^T F_R + g\right) \leq 1. \tag{4.46}$$

Consider the case where $F_R = 0$ (i.e. the external moment is neglected). Eq. (4.46) becomes

$$\tilde{\tau}^T \tilde{\tau} = \left(J_T^T F_T + g\right)^T L_\tau^{-2} \left(J_T^T F_T + g\right) \leq 1, \tag{4.47}$$

which describes the Translational Force Ellipsoid (TFE) in the weak sense. Let $F_T = f\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then Eq. (4.47) can be rewritten as

$$\alpha_T f^2 + 2\beta_T f + \gamma_T \leq 0 \tag{4.48}$$

where

$$\begin{aligned} \alpha_T &= \hat{t}^T J_T L_\tau^{-2} J_T^T \hat{t} \\ \beta_T &= g^T L_\tau^{-2} J_T^T \hat{t} \\ \gamma_T &= g^T L_\tau^{-2} g - 1 \end{aligned} .$$

Solving Eq. (4.48) yields the estimation of the achievable translational EEF force capability given by

$$\frac{-\beta_T - \sqrt{\beta_T^2 - \alpha_T \gamma_T}}{\alpha_T} \leq f \leq \frac{-\beta_T + \sqrt{\beta_T^2 - \alpha_T \gamma_T}}{\alpha_T}. \tag{4.49}$$

The rotational EEF force (moment) capability can be estimated using Eqs. (4.47)-(4.49) with J_T being replaced with J_R.

4.2.5.2 Vector Expansion Formulation

The bounds on τ can be written as

$$-\tau_{\max} \leq \tau \leq \tau_{\max}. \tag{4.50}$$

Combining Eqs. (4.44)-(4.45) with (4.50) yields

$$-\mathbf{1} \le L_\tau^{-1}\left(J_T^T F_T + J_R^T F_R + g\right) \le \mathbf{1} \tag{4.51}$$

where **1** is a vector of *n*-dimension with each element equal to one. Rearranging a few terms gives the governing equation as

$$\tau_{lower} \le L_\tau^{-1} J_T^T F_T + L_\tau^{-1} J_R^T F_R \le \tau_{upper} \tag{4.52}$$

where $\tau_{upper} = \mathbf{1} - L_\tau^{-1} g$ and $\tau_{lower} = -\mathbf{1} - L_\tau^{-1} g$.

Consider the case where F_R = 0 (i.e. the external moment is neglected) and let $F_T = f\hat{t}$ where $\hat{t}$ is the unit vector in the direction of interest in the task space. Then, Eq. (4.52) can be rewritten as

$$\tau_{lower} \le f L_\tau^{-1} J_T^T \hat{t} \le \tau_{upper} \tag{4.53}$$

The vector expansion model describing Eq. (4.53) is illustrated in Figure 4-14.

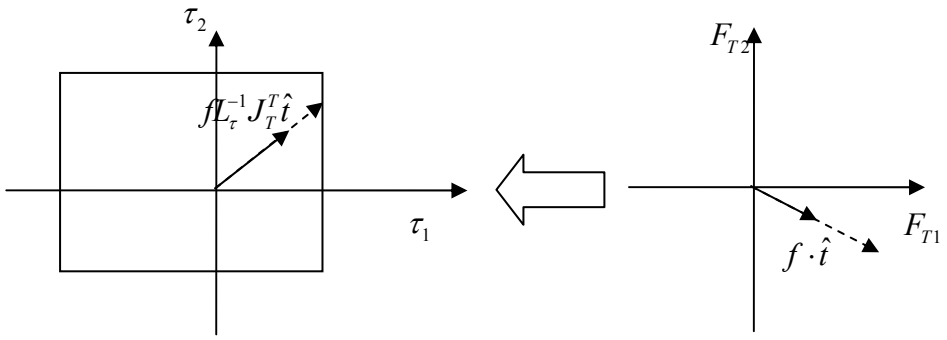


Figure 4-14: Joint torque bounds and force vector expansion.

Note here that, unlike the joint speed bounds, the torque bounds in Figure 4-14 are not symmetrical due to the weight of the robot itself. The largest magnitude of translational force that still fits the transformed force vector in the torque bounds is determined by

$$f \le \min_i \left\{ \max\left(\frac{\tau_{upper,i}}{\left[L_\tau^{-1} J_T^T \hat{t}\right]_i}, \frac{\tau_{lower,i}}{\left[L_\tau^{-1} J_T^T \hat{t}\right]_i} \right) \right\}, \quad i = 1, \ldots, n \tag{4.54}$$

where $\tau_{lower,i}$, $\tau_{upper,i}$, and $\left[L_\tau^{-1}J_T^T\hat{t}\right]_i$ are the i^{th} elements of vectors τ_{lower}, τ_{upper}, and $L_\tau^{-1}J_T^T\hat{t}$, respectively.

Similarly, the rotational EEF force (moment) capability can be found using Eqs. (4.53) and (4.54) with J_T being replace with J_R as

$$m \le \min_i \left\{ \max \left(\frac{\tau_{upper,i}}{\left[L_\tau^{-1}J_R^T\hat{t}\right]_i}, \frac{\tau_{lower,i}}{\left[L_\tau^{-1}J_R^T\hat{t}\right]_i} \right) \right\}, \quad i = 1,\ldots,n. \tag{4.55}$$

Let's reconsider Eq. (4.52) but without assuming that either the force or moment is zero. Let $F_T = f\hat{t}_T$ and $F_R = m\hat{t}_R$ then Eq. (4.52) becomes

$$\tau_{lower} \le fL_\tau^{-1}J_T^T\hat{t}_T + mL_\tau^{-1}J_R^T\hat{t}_R \le \tau_{upper}. \tag{4.56}$$

This equation is illustrated graphically in Figure 4-15. Note that there is no unique solution to Eq. (4.56). One can only find *f* given *m*, or vice versa.

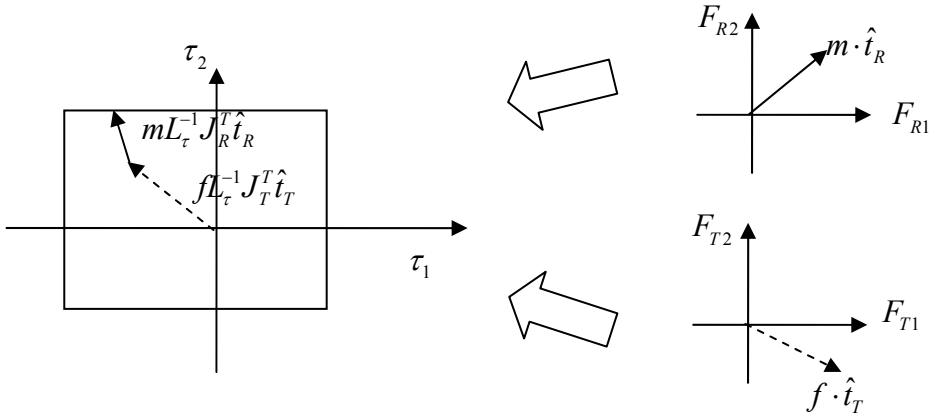


Figure 4-15: Translation and rotation vectors added.

The process of determining *f* given *m* (or vice versa) is straightforward. For a given *m*, Eq. (4.56) can be rewritten as

$$\tilde{\tau}_{lower} \le fL_\tau^{-1}J_T^T\hat{t}_T \le \tilde{\tau}_{upper} \tag{4.57}$$

where $\tilde{\tau}_{lower} = \tau_{lower} - mL_\tau^{-1}J_R^T\hat{t}_R$ and $\tilde{\tau}_{upper} = \tau_{upper} - mL_\tau^{-1}J_R^T\hat{t}_R$. Eq. (4.54) can then be used to determine *f*.

One intriguing application of the estimation of robot force capability is in the area of fault tolerance and failure recovery. Cocca and Tesar [1999] proposed a Partial Failure Torque Minimization criterion for dealing with joint partial failure. Joint partial failure is referred to when one or more joints can no longer supply the torques at their full capacities. The proposed estimation of robot force capability is readily equipped to cope with this circumstance. Once the system detects joint partial failure and determines the reduced torque capabilities for the failed joints, it can set these new values in the joint torque limit matrix L_τ. The new force capability based on the reduced joint torques can then be computed and used in the redundancy resolution process or to create a sense of margin of failure.

To prove the validity of the vector expansion method, we compared the polytope generated by this method with the force polytope algorithm proposed by Chiacchio *et al.* [1997]. Note however that, in the development of the force polytope algorithm, two restrictive assumptions were made: all the joints are of the same kind and the task space of interest is composed by either forces or moments [Chiacchio *et al.*, 1997]. Gravity was also omitted in their formulation. In addition, it was not clear whether or not the force polytope algorithm could provide the limiting joint information. We applied it to a 3-DOF planar robot with the link lengths of 0.5, 0.3, and 0.2 m and with the torque bounds of 3, 2, and 1 N-m. The manipulator configuration is $\theta = \begin{bmatrix} 60 & -60 & -60 \end{bmatrix}^T$. The force polytope generated by the vector expansion method, shown in the figure below, matches the one in Chiacchio *et al.* [1997].

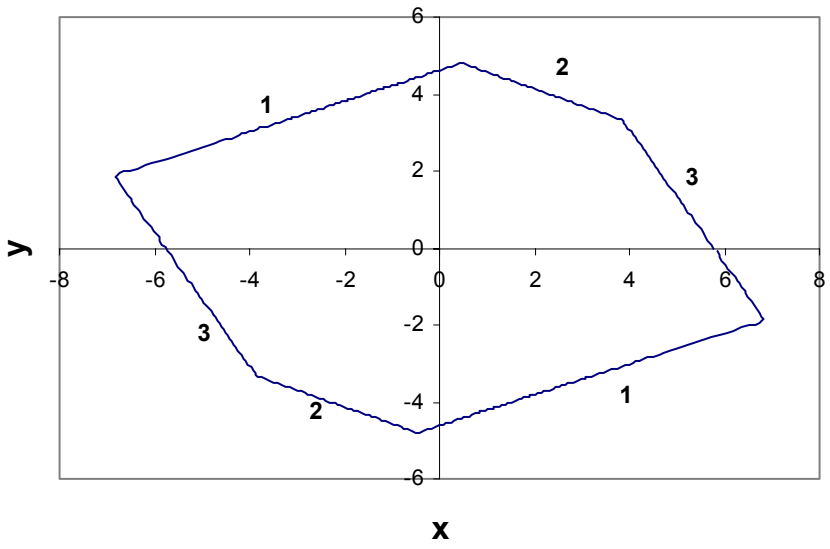


Figure 4-16: Force polytope for a 3-DOF planar robot.

4.2.5.3 Example

Consider the same 4-DOF planar robot again at the same configuration $\theta = \begin{bmatrix} 45 & -45 & -45 & -45 \end{bmatrix}^T$. To illustrate the impact of gravity on the force capability, we consider two cases: with and without gravity. Let the joint torque limit matrix $L_\tau = diag\{100, 45, 35, 15\}$ N-m for both cases. Figure 4-17 and Figure 4-18 show the force ellipsoid (created by the ellipsoid model) and force polytope (created by the vector expansion method) without and with gravity, respectively. It should be noted here that without gravity, both the ellipsoid and polytope are symmetrical around the origin, i.e. the robot can exert an equal amount of force in the direction of interest and the opposite direction (e.g. +X and –X direction). With gravity (assumed to be in the –Y direction), the center of the ellipsoid shifts downwards and to the right and the ellipsoid is no longer symmetrical about the origin. According to the ellipsoid formulation, with gravity, the robot can exert the force downwards almost 3.5 times as much as it can upwards in this particular configuration. The effect of gravity on the polytope is even greater. Not only does

gravity shift the center of the polytope downwards, it alters the shape of the polytope as well. Again, gravity enhances the robot ability to exert the force in the downward direction and reduces that in the upward direction. However, the polytope only shows the ratio of around 2 instead of 3.5 as indicated by the ellipsoid. The significant impact of gravity on the robot force capability is, without question, illustrated by either model.

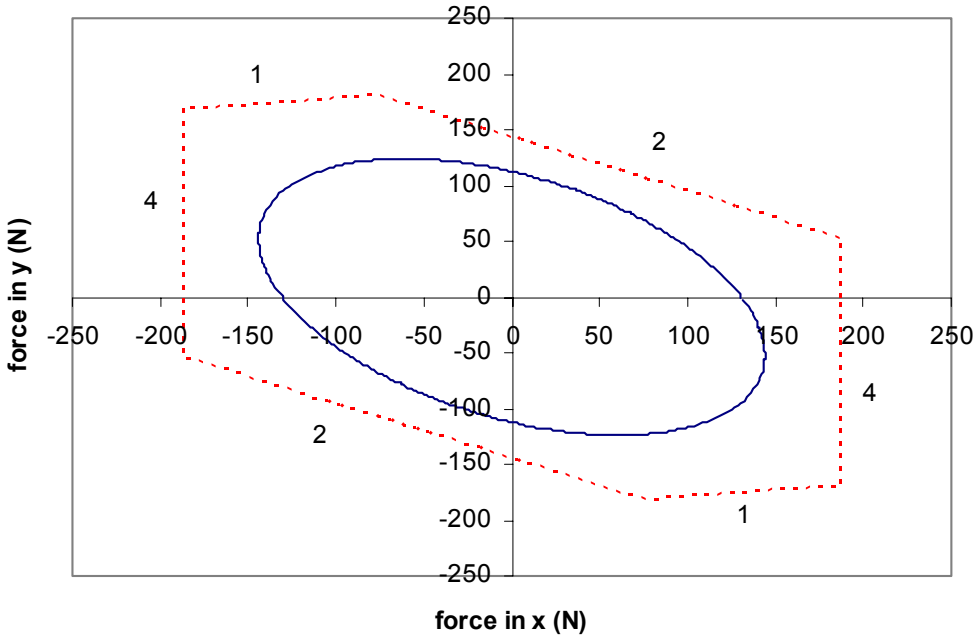


Figure 4-17: Force ellipsoid (solid) and force polytope (dotted) without gravity.

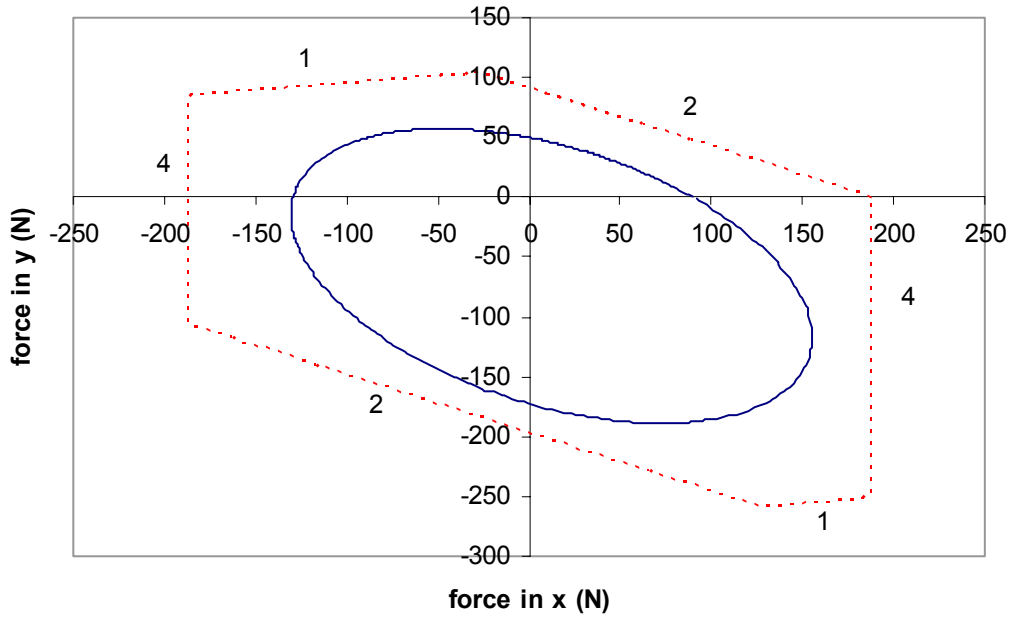


Figure 4-18: Force ellipsoid (solid) and force polytope (dotted) with gravity.

Now if the EEF position is fixed at the point (0.5, 0.0), then using the vector expansion method the minimum and maximum EEF force capabilities in the X direction are found to be 122 N and 244 N, respectively and in the Y direction 90.9 N and 125 N. The corresponding configurations are shown in Figure 4-19.

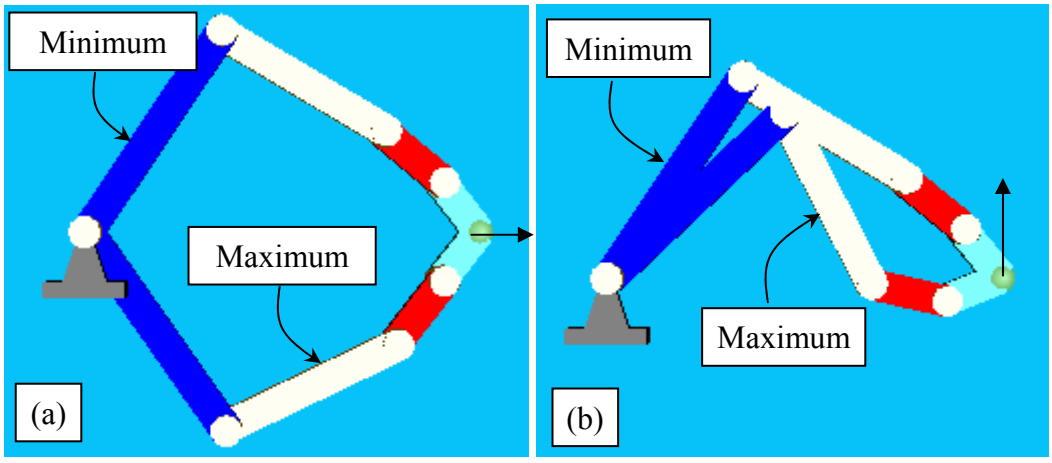


Figure 4-19: The configurations with maximum and minimum EEF froces in (a) +X direction and (b) +Y direction.

4.3 Chapter Summary

The robot capabilities are functions of, among other factors, the joint capabilities and the joint configuration. In this chapter, we presented a new approach to accurately estimating the robot capabilities in the task space called the *vector expansion method*. We developed formulations based on the vector expansion method for each of the robot capabilities (speed, accuracy, and force) that were considered in this chapter. We also gave numerical examples and compared the results with the widely accepted ellipsoid method. Also, the application of this technique to fault tolerance and failure recovery was briefly discussed and demonstrated.

As has been shown, the vector expansion method yields more accurate estimations than the ellipsoid method because it uses the infinity norms of the bounds of the joint capabilities instead of the Euclidean norm used in the ellipsoid method. Using the reverse mapping, the vector expansion method does not require the generation of the whole polytope and as a result is computationally fast enough to be used in a real-time redundancy resolution process.

The robot capability analysis will be of great benefits for the following types of applications.

- **Task-based decision making**

In task-based decision making, robot capability estimations are used in the redundancy resolution process in order to determine configurations (in the null space) that satisfy task requirements and eliminate those that do not. Therefore, a solution trajectory obtained in this manner will ultimately meet the specified task requirements (as much as physically possible), enhancing the user-perceived task execution performance. This is the focus of this report and the formulations developed in this chapter will be utilized in a novel redundancy resolution technique presented in the next chapter.

- **Online and offline path planning**

In online path planning, robot capabilities are continually monitored and measured against task requirement specifications. When one or more robot capabilities diminish and approach the values of associated task requirements, the system can then alert the user or better yet send information to an online path generator so it can adjust the robot path accordingly. Robot capability estimations can also help an offline path planner to generate trajectories that will ensure that task requirements can be met throughout. Note that the use of robot capability analysis to online and offline path planning is equally applicable to both redundant and non-redundant robotic systems.

- **Design and/or assembly of robotic manipulators**

Robot capability analysis can be an invaluable tool in designing robotic manipulators and/or assembling modular ones from modules (e.g. actuators and links). In the design and/or assembly process of a manipulator, a number of target tasks are specified and robot capability analysis (together with other tools) can then be used in an iterative procedure of determining the required joint capability (speed, torque, etc.) limits.

CHAPTER 5

TASK-BASED DECISION MAKING

In Chapter 3, we reviewed in detail traditional redundancy resolution techniques and performance criteria found in the literature. We have seen that traditional redundancy resolution techniques focus on finding a solution that optimizes a performance objective function. The objective function can be a single performance criterion or a composite index resulting from combining multiple criteria.

One major problem of performance criteria is that most of them are based purely on mathematical formulations. Even though UTRRG has made several attempts at understanding the physical meanings of these criteria [Browning and Tesar, 1996] [Pryor and Tesar, 1999] [Tisius *et al.*, 2004], still many of these criteria do not have task-level interpretations that an average user can understand. Another is that even if physical meanings are understood, the chore of selecting and combining criteria for a given task is still daunting, mostly due to couplings and conflicts among criteria. For example, a velocity transmission ratio is a ratio between the magnitude of the EEF velocity in the direction of interest and the norm of the joint velocity vector. Maximizing this criterion means the EEF can move in the desired direction with minimal effort from the actuators. At the same time, maximizing velocity transmission ratio also means that the effect of joint errors is also maximized at the EFF, which may not be desirable. Generalized stiffness is the overall ability of the EEF to withstand a general load without causing too large a deflection. Maximizing this criterion can reduce the deflection at the EEF and therefore improve accuracy. Some questions arise. Should we minimize or maximize the velocity transmission ratio criterion? Under what circumstances should we maximize the generalized stiffness criterion and not the velocity

transmission ratio or vice versa? Or we probably wish to maximize both of these criteria but which one should we give more import?

For constraint-based criteria, optimizing them may not provide any performance advantage. For example, maximizing Joint Range Availability criterion will force all the joints to stay as close to their midpoints as possible. However, we only need the robot to *not* exceed the joint travel limits or get within a prescribed distance from the limits during its operation. There is no performance gain[12] from joints staying near the midpoints as long as the joint travel limits are not exceeded. Redundancy would be better utilized to improve other performance metrics such as load capacity or accuracy as the task requires.

Typically in practice, a set of performance criteria for a given task is chosen by an experienced operator. Assigning weights for all criteria is also done by experience normally through trial and error. Several methods have been recently proposed to assign these weights automatically [McGhee *et al*., 1994] [Hanson and Tolson, 1995] [Cheng *et al.*, 1996 and 1997]. Pryor and Tesar [2002] have introduced a concept of *critical boundaries* to help select active constraint-based criteria. In this scheme, only the criteria that are critical to avoiding system constraints are passed on to the active RRT. The idea behind this strategy is that constraint-based criteria should be included only when necessary. It is a waste of resources for the system to use them to, for example, avoid obstacles when there is no obstacle in the neighborhood. In that case, redundancy would better serve the system if it is used to optimize other criteria. Although the critical boundaries concept automates the process of selecting constraint-based criteria, such as obstacle avoidance, joint limit avoidance, etc., it still leaves a lot to be desired. Consider the following scenario. Suppose the operator wants the robot to be able to exert maximum force in the Y direction. He can certainly try to maximize the force

[12] There is, however, a significant indication of a reserve in performance.

transmission ratio in the Y direction. However, what if the operator wants the robot to be able to exert maximum force in the Y direction and in the mean time achieve a certain level of accuracy tracking in the X direction? He could try to minimize the velocity transmission ratio in the X direction while maximizing the force transmission ratio in the Y direction. However, what value of weight should be assigned to each criterion? What if the accuracy needs to be maintained in the Z direction too? To the best of our knowledge, no work has been done that could address this problem.

In Chapter 4, we provided the definitions of the task requirements and more importantly developed the *vector expansion method* as a tool to accurately estimate the robot capabilities in the task space. In this chapter, we will develop a task-based decision-making scheme for redundant manipulators. The scheme utilizes the improved Generate Options method discussed in Section 3.1.6, the specifications of the task requirements and the robot capability estimations discussed in the previous chapter. Our plan is to attack the problem directly with the help of mathematics underlying the formulations of criteria. *Specifically, a task is described not only by the EEF position trajectory but also by a set of additional task requirements consisting of numerical values of desired speed, EEF force, and accuracy.* Actual robot capabilities at each configuration can be estimated from the robot properties and mathematical models. Then, a configuration that best satisfies the task requirements is chosen. This chapter will utilize them in the redundancy resolution process.

5.1 Task-Based Redundancy Resolution

This section details the development of a novel *Task-Based Redundancy Resolution* (TBRR) method. TBRR is unique in the sense that it revolves around exploiting redundancy to help the robot perform the task at hand successfully. That means, the extra resources must first be used to help the robot perform the task

while complying with system constraints and satisfying speed, accuracy, and force requirements. The remaining resources can then be utilized to minimize the energy consumption or other criteria as desired.

This TBRR concept is illustrated in Figure 5-1[13]. The big oval represents the overall null space at a given configuration. The red (or shaded) regions correspond to *violations* of system constraints, shown here are joint limit, obstacle, and singularity. Examples of violations of system constraints include at least one joint exceeds its travel limits; the robot is colliding with an obstacle; or a mathematical singularity occurs. The diagonal patterned regions denote *violations* of the task constraints[14]. These imply that the robot is not able to satisfy the task requirements. For example, the robot may not be able to apply as much force as required, etc. Therefore only the null space in the middle is deemed acceptable in terms of satisfying the constraints and task requirements. The best configuration according to an efficiency-related or another criterion is then selected from this acceptable null space. In effect, TBRR optimizes only one criterion in a reduced null space as opposed to traditional RRTs that usually optimizes multiple criteria in a full null space.

In terms of avoiding system constraints, TBRR is conceptually similar to the critical boundary concept [Pryor and Tesar, 2002]. Fundamentally, they are drastically different. In the critical boundary concept, the system uses critical boundary information to determine which performance criteria to be *active* (active criteria are included in the optimization of the underlying RRT). In TBRR, constraints can be considered *active* if their associated buffered regions lie in the

---

[13] This figure is exaggerated to present the TBRR concept. In reality it is very rare that violation regions of all constraints (system and task) will be present in the null space. It is much more likely that we will encounter only a few violation regions or none at all in the null space.

[14] We use the terms 'task requirement' and 'task constraint' interchangeably.

null space. Unlike the critical boundary concept, these active constraints are not used to "filter in" necessary performance criteria but instead they are used to "filter out" non-compliant configurations in the null space. The same approach also applies to task requirements, which have no counterparts in the critical boundary concept or any other RRT for that matter.

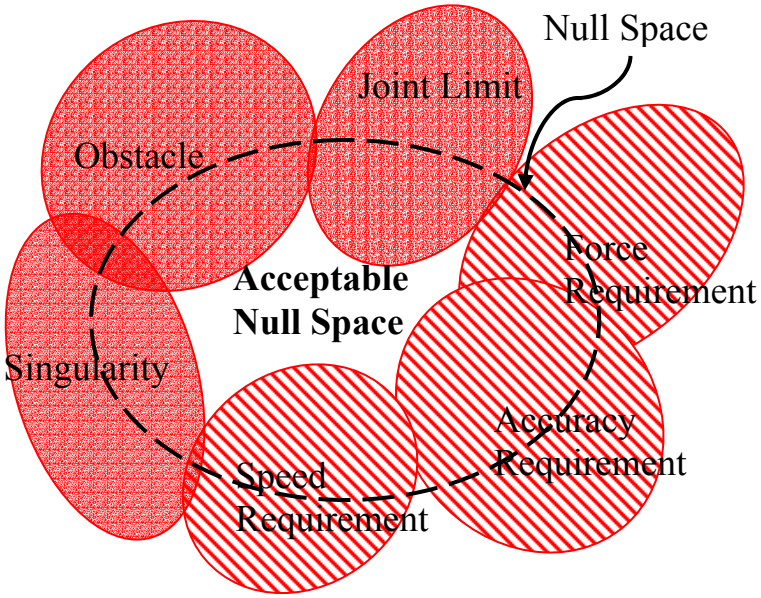


Figure 5-1: TBRR Concept of Constraints and Task Requirements

Conceptually from a task requirement perspective, TBRR is said to attempt to find a solution whose manipulator ellipsoids are larger than the corresponding task ellipsoids[15], which means that the manipulator can satisfy all the task requirements. This is graphically illustrated in Figure 5-2.

---

[15] Although in reality the ellipsoids are not used because we use the vector expansion method (see Chapter 4), this statement is still conceptually valid.

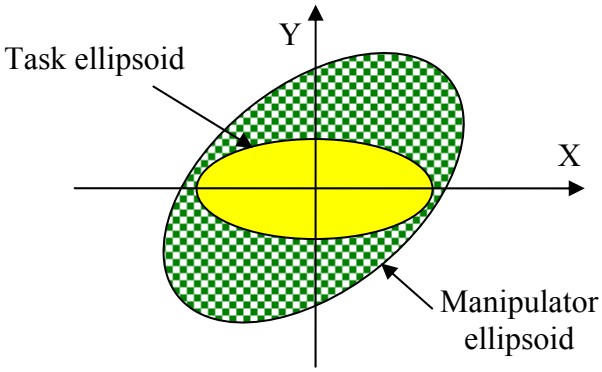


Figure 5-2: Task interpretation of TBRR.

5.1.1 TBRR Scheme

Not only does the Generate Options method have many advantages as stated in Chapter 3, but it is also suitable for TBRR due to the fact that the number of generated options is finite, facilitating the application of *sequential filtering* as a method of filtering out unsatisfactory solutions. Thus the Generate Options method is chosen as the base upon which we will build the TBRR algorithm. Recall from Section 3.1.6, the improved Generate Options method can be summarized as follows:

$$\delta\theta^0 = 0$$
$$\theta^k_{base} = invKin(x^k_d)$$
$$\Theta^k = generateOptions(\theta^k_{base} + \alpha\delta\theta^{k-1}, \Delta\theta)$$
$$\theta^k = pickSolution(\Theta^k)$$
$$\delta\theta^k = \theta^k - \theta^k_{base}$$

The details of all algorithms except *pickSolution* have been previously discussed. Traditionally, *pickSolution* chooses the solution with the best *composite index*. Two popular methods to compute the composite index are the weighted sum of performance criteria and the weighted sum of the rankings of performance criteria. Refer to Kapoor and Tesar [1996] for more detailed discussion of these composite index calculations. This section will present TBRR as a *pickSolution* algorithm.

The flowchart of TBRR is depicted in Figure 5-3. TBRR begins with a set of options that satisfy the EEF constraints. The process of generating options was detailed in Section 3.1.5. Then, a filtering technique similar to the sequential filtering technique [Eschenbach and Tesar, 1969] is applied to these generated options in the solution selection stage. With sequential filters, the number of options to be ranked and computational demands can be greatly reduced. In this application, the sequential filtering works as follows.

1. For each option, the constraint-based criteria are evaluated against the *critical values*[16] specified by the user. The options that do not satisfy all the constraints are *filtered out* and only the options that do are passed on to the next step. The constraints are given priority because of their utmost importance. Constraint violations, at the minimum, will cause the system to fail to complete the desired task. At worse, they could cause damage to the system and/or surroundings.
2. Next, the robot capabilities are estimated and then evaluated against the *desired values*[17] of the task requirements specified by the user. Again, the options that do not satisfy all the task requirements are filtered out.
3. Finally, the best configuration can then be chosen from the remaining options by using one of the criteria for energy minimization discussed in Section 3.2.2.5 or other types of criteria as desired.

In cases where no option satisfies all constraints and/or requirements, conflict resolution is applied. The conflict resolution process is discussed in Section 5.1.3.

---

[16] Critical values are the constraint-based criteria values at the buffer boundaries of the constraints. Buffers are discussed in the next section.

[17] Desired values are task requirement values plus small buffer values. Also, see the next section for details on buffers.

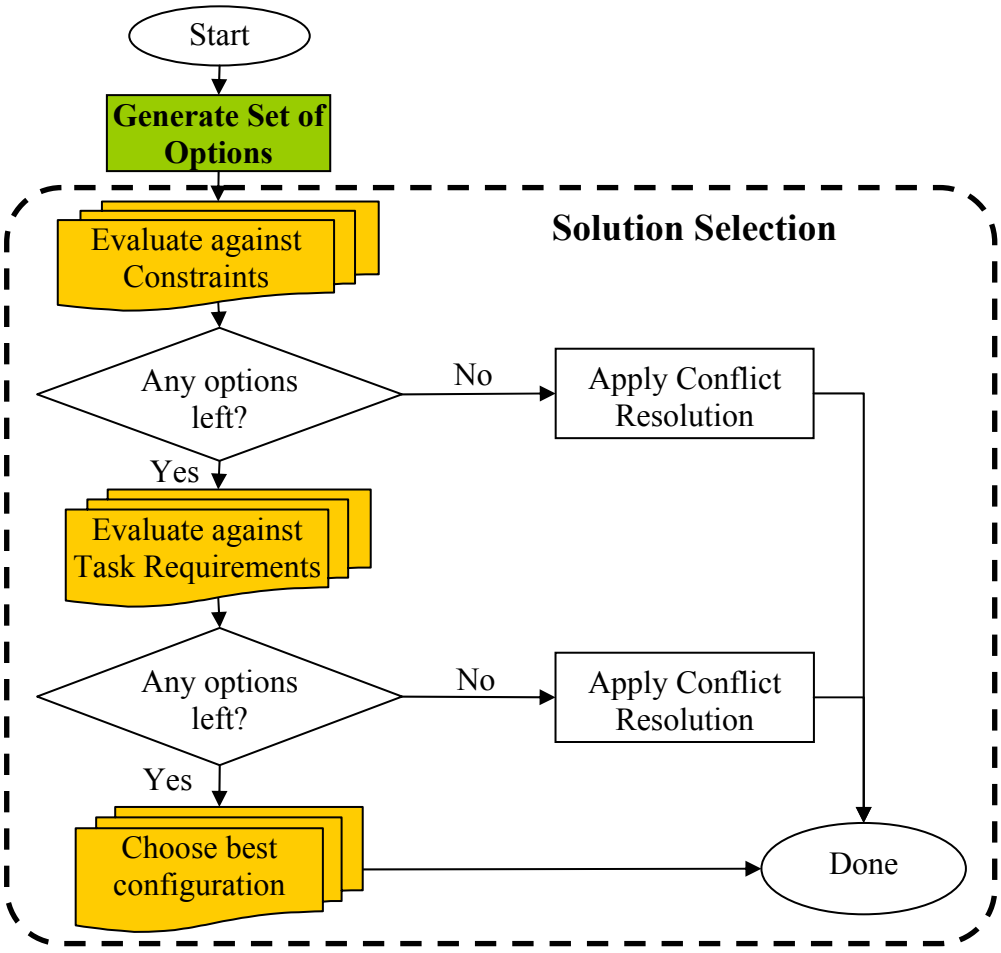


Figure 5-3: Flowchart of TBRR

5.1.2 Buffering

In Figure 5-1, we defined the acceptable null space as the portion of the null space whose configurations do not violate any system or task constraint. If, however, we follow this definition strictly, it could lead to an undesirable result in which the robot operates dangerously close to the edge of violating some constraints. This happens because any configuration that does not violate any constraint will be deemed acceptable even though it may be very close to violating some constraints. If that configuration is chosen, it may drive the robot so close to the violation boundary that the next movement can easily violate the constraints.

As a result, we introduce a concept of *buffering* for all system and task constraints. Here, artificial zones called buffers are placed around violation regions to provide repellence to configurations that are about to enter the violation regions (Figure 5-4). Essentially, buffering is like giving the manipulator a *performance reserve* or an advance warning for each system and task constraint. Configurations in these buffer regions are deemed unacceptable but they are not yet in violation of any constraint. In effect, buffering shrinks the acceptable null space even further. Conceptually from a task requirement perspective, TBRR is said to attempt to find a solution whose manipulator ellipsoids are larger than the corresponding task ellipsoids extended with buffers (Figure 5-5).

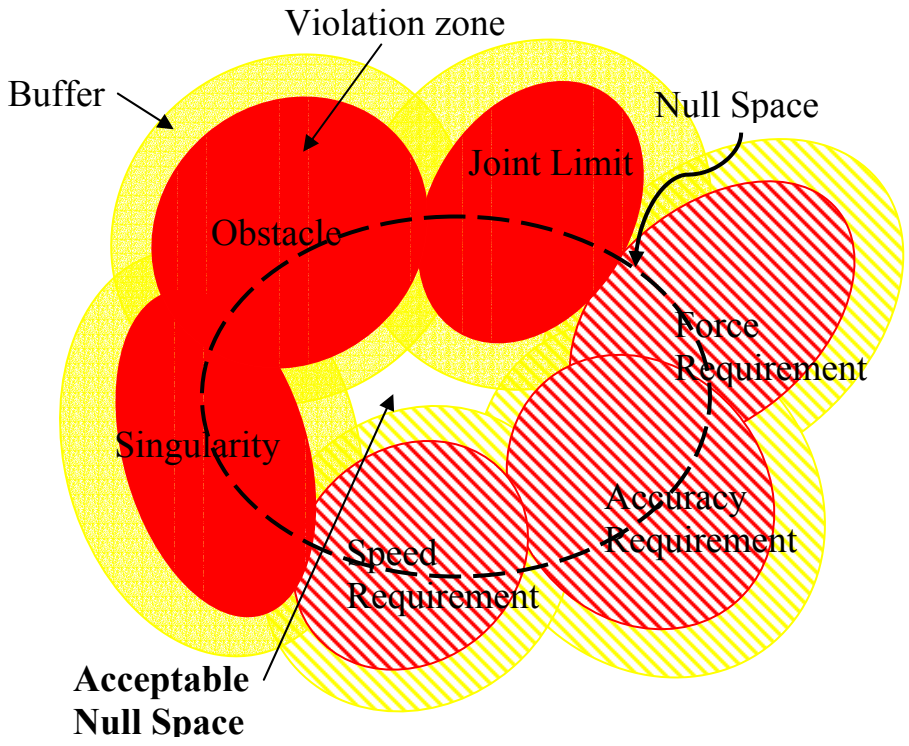


Figure 5-4: TBRR with buffers.

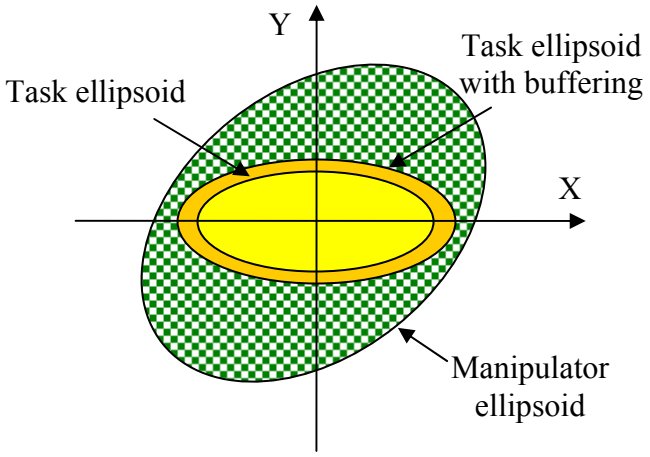


Figure 5-5: Task interpretation of TBRR with buffer.

5.1.3 Conflict Resolution

It is not unusual that none of the generated options is acceptable, i.e. all of them are either in the buffer or violation zone of at least one constraint or task requirement. In that case, two alternatives exist. The first one is to notify the Generate Options scheme to generate a new set of options and apply TBRR again until a limited number of ranked options that satisfy all constraints and task requirements are found. There is, however, no guarantee that one such option exists. It is quite possible that in that neighborhood the robot cannot physically satisfy all the constraints and task requirements (Figure 5-6 and Figure 5-7). This brings us to the second alternative: conflict resolution. In conflict resolution, one would choose the option that *least penetrates* the constraint and/or task requirement buffers. This can be achieved through a concept similar to penalty function in mathematical programming [Luenberger, 1984].

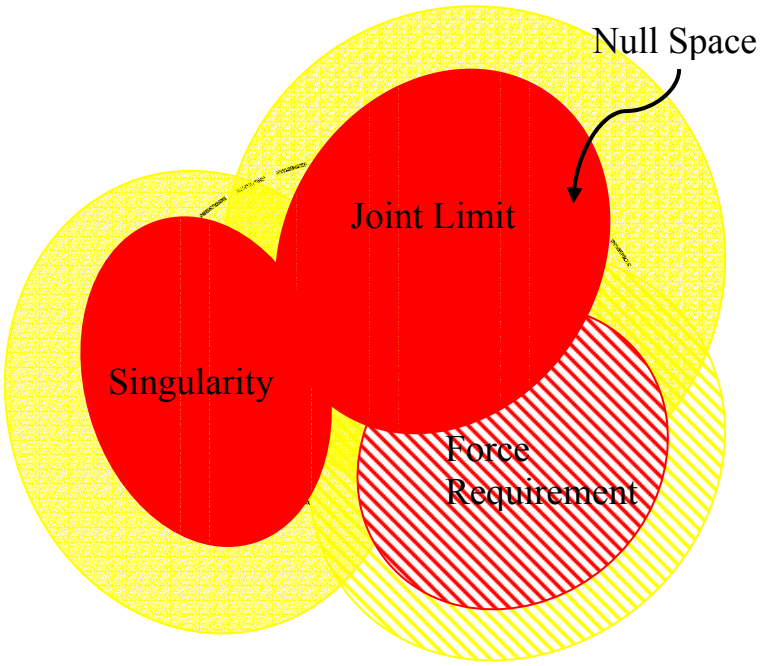


Figure 5-6: TBRR with no acceptable null space, requiring conflict resolution.

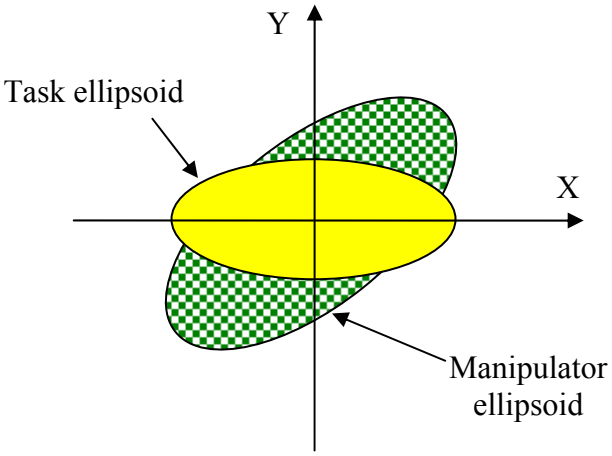


Figure 5-7: Task interpretation of TBRR with conflict.

The concept of conflict resolution of constraints and task requirements is identical. Therefore, we will only show that for the constraints and by analogy the same will be applied for the task requirements as well. For simplicity and without

loss of generality, we assume that all constraint-based criteria are to be maximized. For each constraint, a *deviation* from the critical value is defined as:

$$z_i = \begin{cases} 0; & a_i \geq c_i \\ \left(\dfrac{c_i - a_i}{c_i} \right)^p \times 100\%; & a_i < c_i, c_i > 0 \\ \left(c_i - a_i \right)^p; & a_i < c_i, c_i = 0 \end{cases} \tag{5.1}$$

where a_i is the actual value, c_i is the critical value of constraint i, and p is a positive integer that can be used to amplify the deviation that is far from the critical value. Here if the actual value is greater than the critical value (i.e. the constraint is already satisfied), then the deviation is zero. If not, then the deviation will have some positive value. Then, all the deviations can be added in a linear fashion to form the overall weighted score.

$$Z = \sum_i w_i z_i \tag{5.2}$$

where w_i is the user-selected weight for constraint i. From Eqs. (5.1) and (5.2), if the score is 0, then no constraint is violated. Larger scores mean higher degrees of constraint violation. Therefore, the solution with the lowest score will be chosen.

Normally, all the weights should be set to 1.0 because all the constraints or task requirements (at their respective levels) are considered equally important. However, one could argue that obstacle avoidance is more critical than singularity avoidance since if obstacles are not avoided, it could result in physical damage whereas reaching singularities merely fails the control software. In that case, the weight of the constraint associated with obstacle avoidance can be raised higher.

5.1.4 Specifications of Critical Values and Desired Values

As opposed to traditional RRTs, the user in TBRR is no longer required to choose a set of performance criteria and their relative weights for a given task. The user's job now becomes specifying the critical values of associated constraint-

based criteria and the desired values of speed, force, and accuracy requirements from the task specifications. Therefore the user's burden in TBRR is reduced significantly compared with traditional RRTs.

5.1.4.1 Defining Critical Values for System Constraints

Looking back at the constraint-based criteria presented in Section 3.2.1, this section will focus on how to interpret these criteria in light of determining proper values for the critical values to be used in TBRR.

5.1.4.1.1 Joint Travel Limits

Recall that one of the definitions of the Joint Range Availability (JRA) criterion is (See Section 3.2.2.1)

$$\gamma_{JRA} = \min_i \left(1 - \frac{\left| \theta_i - \theta_{i,\text{mid}} \right|}{\theta_{i,\max}} \right). \tag{5.3}$$

One can see that the JRA criterion value ranges between 0 and 1. The value of 0 signifies that one or more joints have hit their travel limits. The violation value for JRA is then said to be 0. One great thing about this criterion is that it is neither robot- nor unit-dependent. Thus, the guideline given here can be applied to any robot. If the critical value is also set to 0, the constraint will be satisfied as long as no joint has exceeded its limit. This, however, leaves no room for errors. If the critical value is set at 1, then it means that the buffer spans the entire workspace and the constraint is always unsatisfied. This is equivalent to always maximize the JRA criterion. One can say that a value of 0 presents an aggressive strategy whereas 1 implies a conservative operation[18]. The critical value defined this way has a very clear physical meaning. For instance, a critical value of 0.05 means that

[18] The concept of conservative and aggressive strategies are borrowed from the critical boundary concept [Pryor and Tesar, 2002].

the constraint is satisfied if all the joints are within 95% of their ranges. This leaves a reserve of *at least* the remaining 5% for the robot to maneuver its joints without exceeding the joint limits.

5.1.4.1.2 Obstacles

The simplest and perhaps easiest-to-interpret performance criterion for obstacle avoidance is the Smallest Minimum Distance (SMD) criterion discussed in Section 3.2.1.4. The SMD criterion defined in Eq. (3.27) is the smallest value of the minimum distances between all robot links and obstacles. The user can select the critical value to be a safe distance that he thinks is appropriate. For example, a task involving high speed operation is likely to have a larger safe distance than a low-speed task. The safe distance should also be proportional to the robot size. A robot with a reach of 3 meters should have a numerically greater safe distance than a small robot with a reach of 30 centimeters. One can use this critical value directly or in a normalized form. The normalized critical value for the SMD criterion can be defined as

$$\bar{\kappa}_{SMD} = \frac{\kappa_{SMD}}{2 \times reach} \tag{5.4}$$

where κ_{SMD} is the critical value and *reach* is the robot reach. The denominator ($2 \times reach$) presents the minimum distance of concern, i.e. the distance from which it is physically impossible for the robot to collide with an obstacle. The value of 0 for $\bar{\kappa}_{SMD}$ again represents an aggressive mindset because it means that the constraint is satisfied as long as there is no collision. The value of 1 is very conservative since as soon as an obstacle enters the robot workspace, the constraint is unsatisfied.

5.1.4.1.3 Singularity Software Failure

As discussed in Section 3.2.1.5, Measure of Transmissibility (MOT) defined in Eq. (3.3.2) can be used to gauge the relative health of the Jacobian for matrix inversion purposes. The MOT value can range from 0 to a maximum value[19]. This maximum number is not only robot-dependent but also unit-dependent. In practice, there exists a small positive MOT threshold below which the inverse kinematics calculation could fail. This threshold is therefore the violation value for MOT. This threshold is robot- and unit-dependent and can be determined empirically [Tisius *et al.*, 2004]. The critical value for singularity avoidance should be set at little bit higher than this threshold. The normalized critical value for MOT is defined as

$$\bar{\kappa}_{MOT} = \frac{\kappa_{MOT} - MOT_{\min}}{MOT_{\max} - MOT_{\min}} \tag{5.5}$$

where κ_{MOT} is the critical value; $MOT_{\min}$ and $MOT_{\max}$ are the MOT threshold and the maximum MOT of the robot, both of which should be determined empirically. Again, similar to the above two constraints, the values of 0 and 1 for $\bar{\kappa}_{MOT}$ represent aggressive and conservative operations, respectively.

5.1.4.2 Selecting Task Requirement Values

It has been assumed in this study that robotic tasks can be described numerically in terms of desired EEF position, velocity, force, and accuracy. The process of obtaining these numerical values is beyond the scope of this research. However, once these values are attained, they can immediately be passed on to the

---

[19] It is a misconception that MOT ranges from 0 to ∞ while, in fact, MOT is bounded. However, the upper bound is difficult to determine for complex robots with no symbolic Jacobian representations. Nevertheless, it can be determined empirically by moving the robot around the workspace and recording the MOT value.

TBRR algorithm without any modification. We believe that a broad definition of the task requirements of EEF speed, force, and accuracy can accommodate a majority of today's robotic tasks.

As for selecting the desired values of the task requirements, one may add buffers to the task requirement values. The sizes of the buffers to be added depend on the nature of the task at hand. For example, let's say a robot is to perform a saw-cutting task with the force requirement of 100 N. When the robot is moving in free space to the cutting piece, there is obviously no force requirement. However, to prepare the robot for the cutting operation, a force buffer of 100 N (equal to the requirement) or more should be added when the robot is approaching the cutting piece. While cutting, we may also add a buffer of say 10 N and pass the desired value of 110 N onto TBRR. This will allow the robot to adjust its configuration when its estimated force capability comes down toward 110 N, instead of 100 N, providing some maneuvering room.

5.2 Subjective Parameters

An excessive number of system parameters in RRTs and the subjectivity of these parameters are major contributors to the complexity of deploying redundant manipulators. Common for all RRTs that employ the weighted sum of criteria, the most prevalent subjective parameters are certainly the weights themselves. Other subjective parameters are usually specific to the underlying optimization schemes or search methods. For instance, the factor k in Eq. (3.6), which specifies how much the gradient of the performance objective should influence the outcome of the gradient projection method, is a subjective parameter. So are the maximum allowable variations (M_j in Eq. (3.9)) in the compromise solutions method.

As typical with any other RRT, TBRR is founded on a strong mathematical and analytical ground, which usually requires precise mathematical models, relationships, and/or formulations. However, the world is not ideal and these

precise models or relationships can seldom be obtained. For instance, there are several parameters in TBRR or any other RRT. It is almost impossible to mathematically express the relationships of these parameters to the overall performance of the system. Usually, these relationships are embedded in the form of human knowledge on the underlying methodologies. It is not uncommon, for example, to hear that if we increase the value of parameter A, then we should obtain more accurate solution but it may cause computational instability if the value is raised too high. If the relationships between the system parameters and the system behaviors could be obtained as mathematical expressions, then it would be possible to use some kind of optimization techniques to solve for optimal parameter values. However, this is rarely the case, if ever. Nonetheless, this problem has not kept the human operator from obtaining reasonably good values for system parameters. The human operator achieves this by trial and error. The trial and error technique is a way the human incorporates lessons learned from previous experience to help improve future performance of the system.

Like any other RRT, TBRR also has several subjective parameters as listed in Table 5-1. Our experience shows that changes in these parameter values affect the task performance in one way or another. It is interesting to see how we can use lessons learned to help determine proper values of these parameters. In this pilot study, we shall limit our discussion only to buffer sizes.

TBRR Components	Subjective Parameters	References
Improved Generate Options Scheme	▪ Search pattern & perturbed joints ▪ Perturbation size & Momentum term	▪ Sections 3.1.5 and 3.1.6 ▪ Eq. (3.11)
Solution Selection	▪ Buffer sizes	▪ Sections 5.1.1 & 5.1.2

Table 5-1: TBRR and subjective parameters.

5.2.1 Buffer Sizes

As discussed in Section 5.1.2, buffers define the critical and desired values for system constraints and task requirements, respectively. These values are passed onto the TBRR algorithm and used to filter out non-compliant options. With buffering, each constraint/requirement can be defined on a scale between aggressive (no buffer) and conservative (large buffer size). Let's look at system constraints, for example. At one extreme, aggressive operation means that only configurations that violate the constraints are filtered out so there are likely more options remaining for the system to use them to maximize task performance. At the other extreme, conservative operation implies that all options are used to avoid the constraints, i.e. no option is filtered out so there is no option left for satisfying task requirements. However, this does not necessarily imply that none of the task requirements will be satisfied. It only means that the redundant resources are not allocated to satisfy task requirements. If task requirements are not very demanding, then there is a great chance that they will still be satisfied without the help of redundancy.

The operator now has an easy and intuitive decision to make for each constraint/requirement. He can be as aggressive or conservative as he wishes. The ultimate goal of choosing proper buffer sizes is for all the constraints and requirements to be satisfied for the entire task path. If we choose to be too conservative for some constraints or requirements, then it could leave others unsatisfied. If, on the other hand, the aggressive approach is taken for some constraints or requirements, then those constraints or requirements themselves may not be satisfied. So the decision must be made to balance all the buffer sizes. To make the decision more complicated, other factors also play a role in the system performance. Among these factors are:

- Couplings among constraints and task requirements. For example, avoiding singularities could increase the force capability in some places in the workspace while decreasing it in others.
- Local optimization[20]. TBRR and almost every other RRT only perform local, not global, optimization. It has been shown that local optimization does not generally translate to global optimization. In fact, local minimization of some criteria can counterintuitively lead to those criteria values blowing up in the global sense. For example, minimizing joint velocity norm causes the robot to approach singularities [Carignan, 1991]. Joint torque minimization leads to system instability in a long trajectory [Hollerbach and Suh, 1985].

We must note that for most simple, undemanding tasks, reasonably small buffer sizes for all constraints/requirements should suffice. What is more interesting is how TBRR performs in complex and demanding tasks and how we can hand-tune the buffer sizes in order for TBRR to perform well.

Demonstrations in the next chapter will illustrate that the buffer sizes can be hand-tuned to improve the system performance. However, this process is tedious and time-consuming. The next section presents an alternative to hand-tuning the buffer sizes by using machine learning to automate the search of good buffer sizes.

5.2.2 Learning Buffer Sizes

Trial and error approach, which is used by the human operator for hand-tuning system parameters, is tedious and time-consuming, especially in a high-dimensional parameter space. In trial and error, starting with an initial set of

---

[20] Local optimization in this context differs from the same term in the general mathematical programming context. Local optimization here means that the solution is optimized at an instant in the path instead of the entire path.

parameters, the operator *executes* the task and *observes* the results. If the results are unsatisfactory, he then makes appropriate changes to the parameter values according to his experience and understanding of the system, executes the task, and again observes the results. He repeats this process until he is satisfied with the results or runs out of time or patience. In this section, we wish to automate the process of obtaining a good set of system parameters, focusing on the buffer sizes. Given a total of N buffers, our task here amounts to a parameter optimization in a continuous N-dimensional space. The learning method presented here is similar to policy gradient reinforcement learning used by Kohl and Stone [2004] to learn a fast walk by the Sony Aibo robot. There are other minor differences but the main one is that the learning method here does not use gradients for a reason to be stated later.

Our approach is described in the following steps:

1. Let the initial state $\pi = \{b_1, b_2, \ldots, b_N\}$, where N is the total number of buffer sizes to be learned, be the best state;
2. Evaluate the best state. Here we let the robot execute the task and record its performance. The performance index is then defined as:

$$p = \sum_{i=1}^{C} wc_i + \sum_{j=1}^{R} r_j \tag{5.6}$$

where c_i and r_j are the numbers of step points in which the i^{th}-constraint and j^{th}-requirement are satisfied, respectively; w is a positive large number to give more importance constraint violations; C and R are respectively the total number of constraints and task requirements. Therefore, we want to minimize the performance index p. Note that p is discrete since the number of step points can only be integers and if w is large, the value of p can jump considerably even with a small change in c_i. Thus any gradient-based learning algorithm would not work.

If the best state satisfies some termination criterion, stop. Among termination criteria are:

a. All constraints and task requirements are satisfied ($p = 0$).
b. Performance has not improved for a number of iterations.
c. A pre-determined number of iterations has been reached.

3. Generate t new states or policies $\{\pi_1, \pi_2, \ldots, \pi_t\}$ from the best state. This step is equivalent to the human operator making changes to the parameter values. Here, we generate random policies $\{\pi_1, \pi_2, \ldots, \pi_t\}$ near π, such that each $\pi_i = \{b_1 + \Delta_1, b_2 + \Delta_2, \ldots, b_N + \Delta_N\}$ and each Δ_j is *probabilistically* chosen to be either $+\varepsilon_j$, 0, or $-\varepsilon_j$, where ε_j is a small positive value for each b_j. From understanding of TBRR, we know that if constraint j is not satisfied throughout the entire path ($c_j > 0$), then we should *probably* increase the corresponding buffer size b_j. However, due to other factors as discussed in the previous section, we should not ignore the possibility that decreasing the buffer size could also improve the system performance. Therefore,

   - if c_j or $r_j > 0$, then the probabilities of Δ_j being either $+\varepsilon_j$, 0, or $-\varepsilon_j$ are 0.5, 0.25, and 0.25, respectively.
   - if c_j or $r_j = 0$, then all three probabilities are equal at $\frac{1}{3}$.

   One can choose ε_j to be a small fixed value or start ε_j with a large value and diminish gradually as iterations go on in the form of

$$\varepsilon_j = \left(\varepsilon_{\max,j} - \varepsilon_{\min,j}\right) e^{-\eta k} + \varepsilon_{\min,j} \tag{5.7}$$

where k is the number of the current iteration, η a decaying factor, and $\varepsilon_{\max,j}$ and $\varepsilon_{\min,j}$ the maximum (at the beginning) and the minimum values

of ε_j, respectively. One benefit of varying ε_j is that the solution should converge faster to the optimum.

4. Evaluate the policies generated in Step 3. Here, we can use one of the following heuristic strategies.
    a. *Hill climbing*: Evaluate all the generated policies and select the one with the best performance index.
    b. *Opportunistic*: Evaluate one policy at a time and select the new state as soon as one with better performance index than the current state is found. This strategy is speedier than the hill climbing method.

    If no better state is found, then keep the current state as the best state.
5. Update the best state and go to Step 2.

The parameter learning method proposed here will help eliminate the need for guesswork of the buffer sizes by the user and thus making TBRR even easier to use. In the next chapter, we will demonstrate the effectiveness of this learning algorithm.

5.3 OSCAR Implementation

This section presents the software implementation of TBRR within the OSCAR framework. Recall from Figure 3-4 that ***Fusion*** is the component that performs *pickSolution* in the Generate Options framework. Therefore, we will derive the ***TBFusion*** class from ***Fusion*** to implement our TBRR algorithm. Figure 5-8 shows the class diagram of ***TBFusion*** and its relationships with other classes.

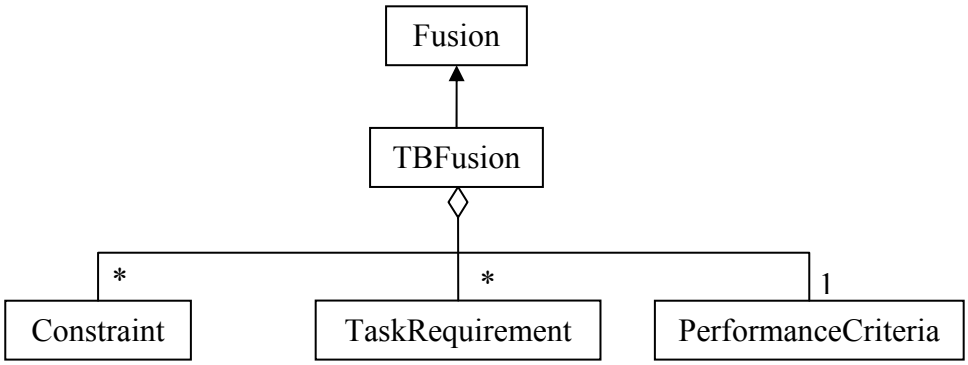


Figure 5-8: Class Diagram of TBRR.

TBFusion consists of three primary components: ***Constraint***, ***TaskRequirement***, and ***PerformanceCriteria***. The multiplicity *'s in Figure 5-8 above these first two components indicate that ***TBFusion*** can contain zero or more objects of these classes. For example, ***TBFusion*** can contain the joint travel limit and singularity avoidance constraints and no task requirement. The multiplicity 1 indicates that ***TBFusion*** must have exactly one ***PerformanceCriteria*** object. The main function of ***TBFusion*** is *EvaluateOptions(options)*, where the argument *options* is the set of generated options obtained from the GO method. *EvaluateOptions* implements the TBRR algorithm outlined in Figure 5-3.

Constraint is constructed from a constraint-based criterion. The primary method of ***Constraint*** is *Compute(robot info)*, which computes the actual constraint value and compare it against the critical constraint value specified by the user via *SetCriticalValue(critical value)* method. ***PerformanceCriteria***, an existing OSCAR class, is an abstract class that provides functionality that is common to the development of specific performance criteria. Any performance criterion can be used in the TBRR framework although an efficiency-related criterion is recommended. Another useful method in ***Constraint*** is *SetActive(true or false)*. Here if the user decides that a certain constraint is no longer of interest, he can easily turn it off by calling *SetActive()* method with the false argument. Similarly he can turn it back on whenever he wishes.

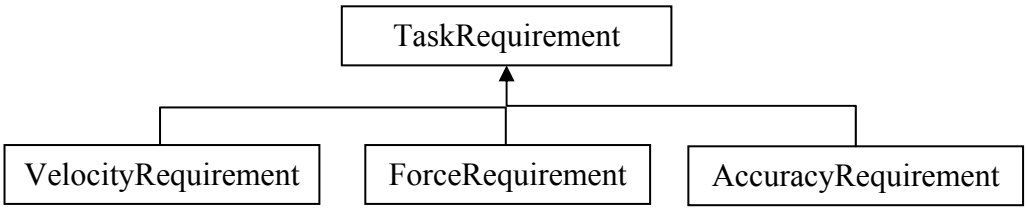


Figure 5-9: Class Hierarchy of Task Requirement Components.

TaskRequirement is an abstract base class for all task requirement classes. It provides the common interface so that all the task requirement classes interact with other classes in the same, predictable manner. The main method of ***TaskRequirement*** is *Compute(robot info)*, which estimates the robot capabilities and compares it against the desired task requirement values specified by the user via *SetDesiredValues(desired values)* method. Note that the user must specify the desired task requirement values for all output directions. The user, nonetheless, can set the desired values to zeros (or very large values in case of EEF position errors) in the directions that do not matter to the task operation. As with ***Constraint***, ***TaskRequirement*** also has a method called *SetActive(true or false)* that lets the user enable or disable a certain task requirement whenever he desires. ***VelocityRequirement***, ***ForceRequirement***, and ***Accuracy-Requirement***, whose names are self-explanatory, are derived from ***TaskRequirement***. Currently, their implementations follow the formulations given in Chapter 4. As has been pointed out, estimations of the robot capabilities based on the concept of polytopes are more accurate but also more complex and far more computationally expensive than those based on ellipsoids. It is conceivable that in the near future as computing power increases, one may wish to use polytopes in estimating the robot's capabilities instead of ellipsoids. It is imperative that our software design would permit such a change without affecting the overall TBRR scheme. Our modular design of ***TaskRequirement*** achieves this ability to change.

5.4 Chapter Summary

In this chapter, we have presented a novel redundancy resolution technique called Task-Based Redundancy Resolution (TBRR). TBRR considers the task at hand as an integral part when it resolves the inverse kinematics. No longer does the user dread combining performance criteria to achieve the desired performance. All the operator has to do is specify the task requirements in terms of desired speed, accuracy, and/or force capability. The TBRR method will then attempt to find a solution that complies with the system constraints and satisfies those task requirements as much as possible, taking into account the robot properties such as kinematic, dynamic, compliance, and actuators' capability limits (speed limits and torque limits) by way of the Vector Expansion method developed in the previous chapter. TBRR uses Generate Options as the method to search through the null space and then applies the sequential filtering technique to sift out non-compliant solutions. The best ranked solution is then easily chosen from a user-defined criterion such as efficiency, etc. The concepts of buffering and conflict resolution were also presented as parts of TBRR overall scheme in order for TBRR to work more effectively.

The software implementation of TBRR adheres to OSCAR specifications of generality, extensibility, reusability, and modularity. This means that TBRR can be applied to any serial manipulator. In addition, the task requirement modules can be easily added or their implementations be replaced with more accurate estimations of robot capabilities when they become available and/or computationally viable.

Finally, an issue was raised about parameter subjectivity, i.e. difficulty in choosing proper values of system parameters of RRTs in general and TBRR in particular. The buffer sizes were then discussed in details and an algorithm to automate the selection of buffer sizes was proposed. This algorithm was intended to replace the trial and error approach normally utilized by the human operator.

The next chapter will demonstrate the effectiveness of TBRR method on a number of computer-simulated robots performing a variety of tasks. Also, the parameter subjectivity issue will be illustrated in simulations and results from the learning algorithm presented.

CHAPTER 6

TBRR DEMONSTRATIONS

The purposes of this chapter are two fold. In the first half of the chapter, we discuss the development and results of the simulations in order to demonstrate the task-based decision making approach presented in the previous chapter. The second part of the chapter is devoted to demonstrating the parameter learning of buffer sizes using the method described in the previous chapter.

6.1 Demonstrations

In this section, we examine the performance and applicability of the proposed scheme by computer simulations in three demonstrations. The first demonstration compares the effectiveness of TBRR with traditional RRTs on a 10-DOF robot. The second illustrates the generality and versatility of TBRR by applying it to two geometrically different manipulators performing the same task. Two manipulators were chosen in these simulations. The final demonstration makes a first attempt to use TBRR on a traditionally non-redundant manipulator.

6.1.1 System Descriptions

Before going into details of all the planned demonstrations, let's first introduce the three robots that will be used in these demonstrations. The first manipulator is a classic 6-DOF UNIMATE PUMA 760 robot (hereafter referred to as PUMA) shown in Figure 6-1. It has a reach of around 1.3 m. Figure 6-2 depicts the second robot which is a 7-DOF Mitsubishi PA10-7CE robot (hereafter referred to as PA107). The PA107 robot is known for its compact, light-weight design [Mitsubishi Heavy Industries, Ltd.]. Its reach of 930 mm is comparable to that of

human arms and it weighs only 40 kg. The necessary properties of the these robots, including D-H parameters, link masses and inertias, etc. can be found in Appendix A. The last robot is a conceptual 10-DOF Fault Tolerant robot (Figure 6-3). This robot will be hereafter referred to as FT10. It has a reach of about 2.2 m. Because of its high degrees of redundancy, it has been used in previous research work by Cetin and Tesar [1999] and Pryor and Tesar [2002].

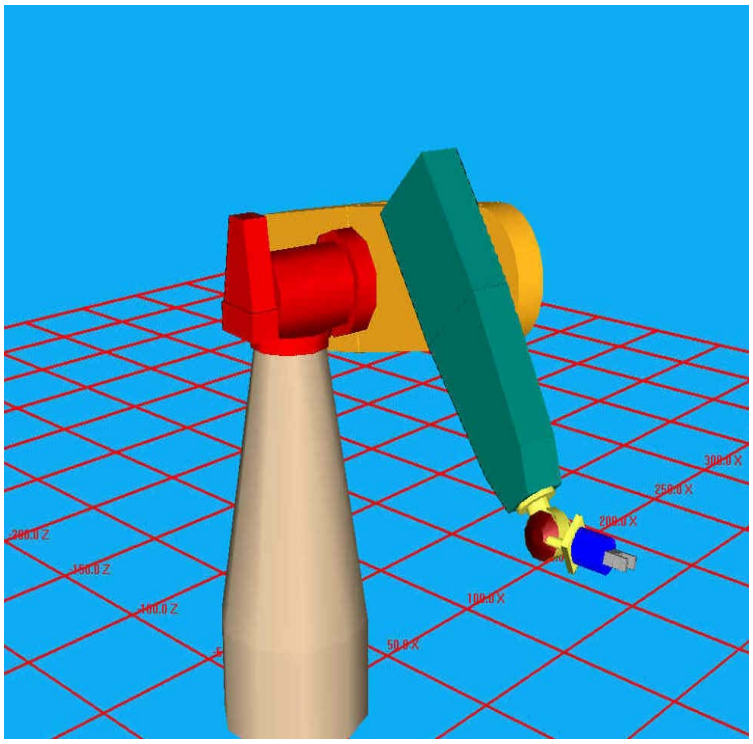


Figure 6-1: The PUMA 760 robot.

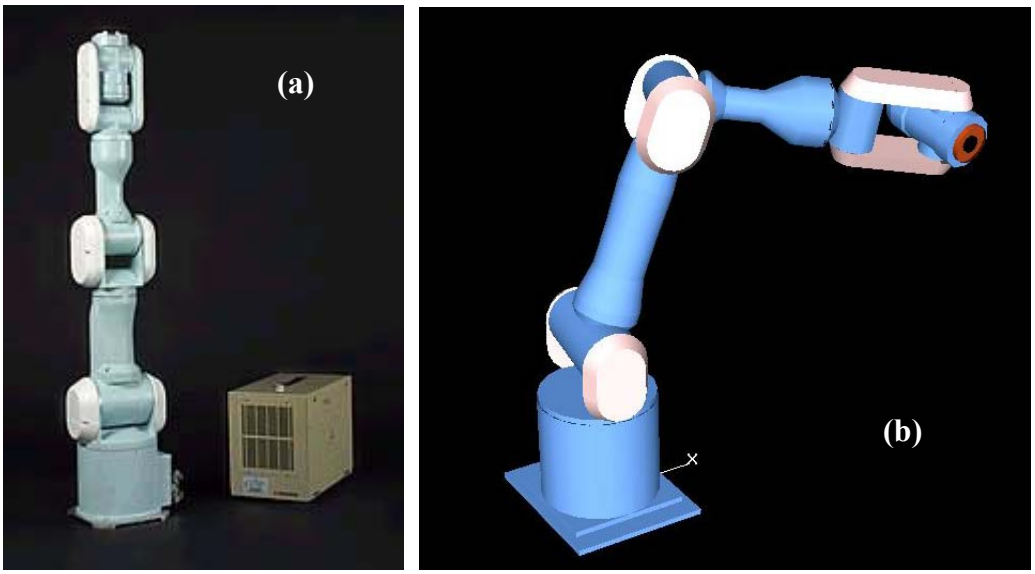


Figure 6-2: (a) Actual Mitsubishi PA10-7CE robot with controller (b) Graphically simulated robot.

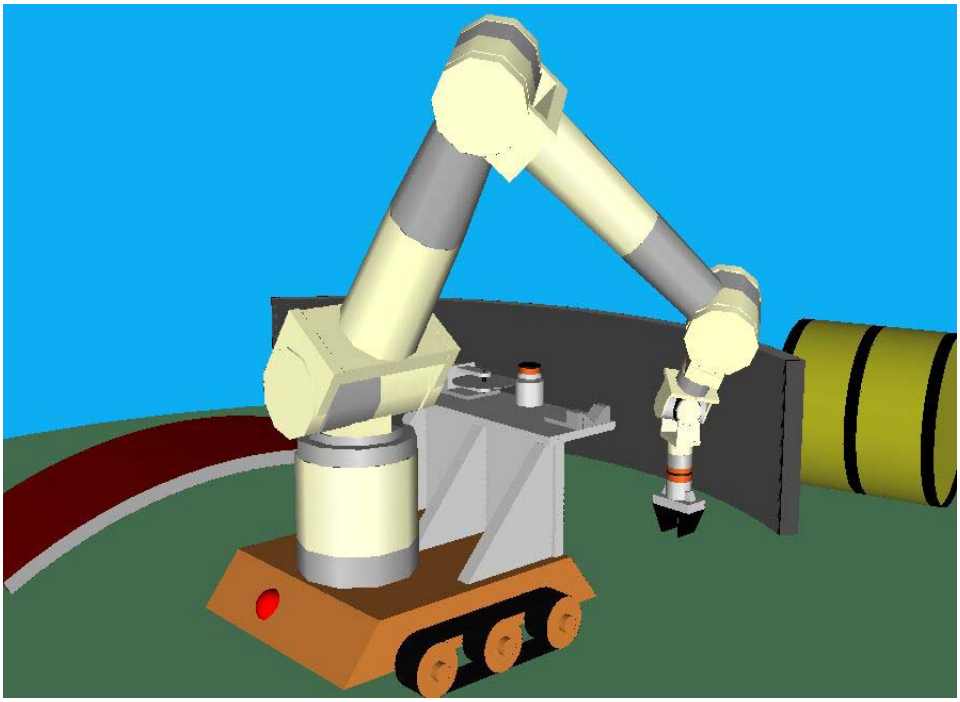


Figure 6-3: The FT10 Robot.

6.1.2 Comparative Study on a 10-DOF Manipulator

To demonstrate the effectiveness of TBRR, we will run simulations on the FT10 robot using traditional RRTs and TBRR and compare the results. The FT10 robot was chosen due to its expanded set of resources. The RRTs chosen in this comparative study were the pseudoinverse, the improved Generate Options (GO) discussed in Section 3.1.6, and GO with critical boundaries [Pryor and Tesar, 2002]. The pseudoinverse is included for baseline comparisons. The reason we chose the other two methods is because we want to comparatively evaluate the effectiveness of TBRR and not the intricacy of the underlying RRT. Since both TBRR and GO with critical boundaries use GO to search the null space, the only difference amongst the three methods is in how they pick the solution from the solution space.

In these simulations, we will also see the motivation behind the development of TBRR. It will be shown that different performance criteria are coupled and selecting appropriate criteria for a given set of task requirements is a difficult task. For example, optimizing JRA could improve accuracy or force capability or maximizing MOT could deteriorate the stiffness criteria, which in turn leads to declined accuracy. These couplings are extremely difficult to determine. Their existence has been acknowledged but never adequately addressed.

The task chosen here is a path tracking task similar to what was used by Cetin and Tesar [1999] in their demonstrations of the Compromise Solution method. Here, the robot's EEF is to follow the circular path which is the lid of the barrel shown in Figure 6-4 while also applying a certain amount of force in the –Z direction. The path is trapezoidal at the velocity level and is divided into a total of 600 step points.

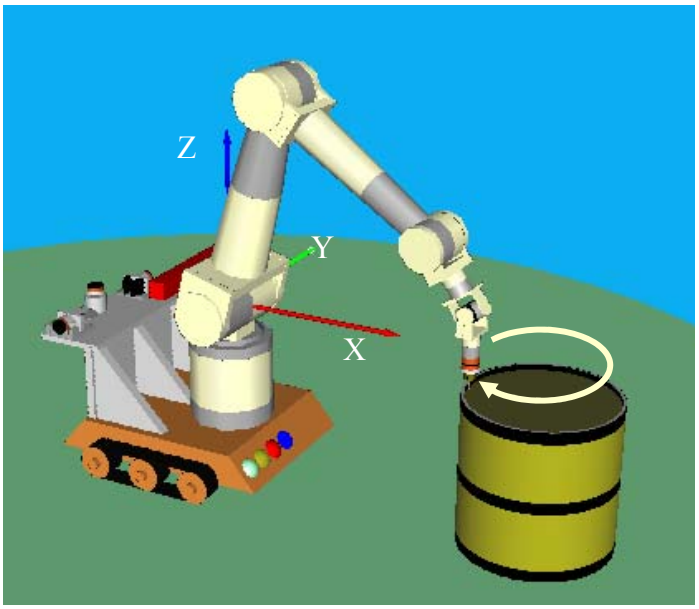


Figure 6-4: FT10 manipulator performing the tracking task.

As previously stated, the four methods used in the simulations are the pseudoinverse (hereafter referred to as PI), the improved Generate Options (GO), GO with Critical Boundaries (CB), and TBRR (TB). We cannot include additional performance objectives in PI. For GO and CB, the following performance criteria are used:

- Joint Range Availability (JRA) defined in Eq. (3.14),
- Measure of Transmissibility (MOT) defined in Eq. (3.32),
- Dexterity (DEX) defined in Eq. (3.33),
- Generalized Load Stiffness (GLS) defined in Eq. (3.51),
- Torque Efficiency (TEF) which is the reciprocal of the FTR criterion defined in Eq. (3.48) with the vector u being the Z direction.

The first two criteria (JRA and MOT) are constraint-related and have critical boundaries associated with them (when used in CB). The last three criteria

are chosen in order to help satisfy the task requirements[21]. The perturbation size $\Delta\theta$ and the momentum term α (see Eq. (3.11)) used in the simulations are 0.2 degree and 0.8, respectively, for GO, CB, and TBRR methods. We used the simple search pattern on all 10 joints and thus the total number of options is 21.

In the experiments to be followed, we ran GO and CB with different sets of weights. Let w_1, w_2, w_3, w_4, and w_5 be the weights of JRA, MOT, DEX, GLS, and TEF criteria, respectively. Since these weights can assume any real positive values, possible combinations of these weights are endless. As previously stated, assigning weights is a matter of experience and trial and error. Here is the list of the sets of weights for GO used in the experiments.

	w_1	w_2	w_3	w_4	w_5
GO1	1	1	1	1	1
GO2	1	1	1	0	0
GO3	1	1	0	1	0
GO4	1	1	0	0	1
GO5	1	1	0	0	0

Table 6-1: Set of weights for GO.

For CB, the weights of the criteria depend on the critical boundaries. When a critical boundary is crossed, the weight of its associated performance criterion is set to 1 and for untriggered criteria, the weights are left at zero. For example, if JRA is critical, then w_1 is 1 and w_2, w_3, w_4, and w_5 are zero. The same is true for MOT as well. If both boundaries are crossed, then both w_1 and w_2 are set to 1 and the other three weights are set to 0. When no boundary is critical, however, we are

[21] Naturally, one could select another set of criteria to be used in satisfying the task requirements that could prove to be better or worse than the three criteria chosen above. The goal of developing an objective and systematic process of criteria selection is one of the main motivations behind this research.

free to assign any values to the weights of DEX, GLS, and TEF criteria. Table 6-2 lists the sets of weights used in the simulations.

	w_3	w_4	w_5
CB1	1	1	1
CB2	1	0	0
CB3	0	1	0
CB4	0	0	1

Table 6-2: Set of weights for CB.

6.1.2.1 Experiment I

In the first experiment, the task requirements in terms of speed, force, and accuracy are set to the values listed in Table 6-3.

	Max. Speed (m/s; deg/s)	**Force (N; N-m)**	**Accuracy (mm; deg)**
X	0.33	0	1.2
Y	0.33	0	1.2
Z	0	300	1.2
Roll	0	0	2
Pitch	0	0	2
Yaw	0	0	2

Table 6-3: Task requirements for Experiment I.

The simulation results are summarized in Table 6-4. The first column designates the method used in the simulation. This second column indicates whether or not the joint limit constraint has been violated (Yes or No). The numbers in the next two columns represent the numbers of step points (out of 600) in which a certain task requirement is not satisfied. Thus, smaller numbers mean better performance with 0 being the best (i.e. the task requirement is satisfied at every step point). Other task requirements not listed in Table 6-4 are satisfied at all step points and therefore need not be compared. The *total* (in the fifth column) is merely a sum of the numbers in the previous two columns. It represents the total number of task requirement violations for a given method along the trajectory. If

all task requirements are considered equally important, then the total can be used as a single performance metric, indicating how well each method performs. One exception is when any constraint violation occurs. Since constraints have higher priority, a method that produces constraint violation will be deemed a failure and inferior to a method without constraint violation regardless of the total number. This total can also be used as a cost function in a learning algorithm that will be discussed in a later chapter. TBRR1 in the table denotes that TBRR was used without buffering, i.e. the task requirement values in Table 6-3 were directly passed on to TBRR. TBRR2 adds a buffer of 20 N for the force requirement. These results will be analyzed and conclusions drawn later after the results of Experiment II are presented.

Method	Joint Limit Violation	Force Violations	Accuracy Violations	Total
PI	***Y***	*600*	*127*	***727***
GO1	***Y***	*201*	*0*	***201***
GO2	***Y***	*600*	*139*	***739***
GO3	**N**	247	0	**247**
GO4	**N**	150	0	**150**
GO5	**N**	316	0	**316**
CB1	**N**	600	133	**733**
CB2	**N**	600	138	**738**
CB3	**N**	600	103	**703**
CB4	**N**	155	0	**155**
TBRR1	**N**	61	0	**61**
TBRR2	**N**	61	0	**61**

Table 6-4: Number of step points in which task requirements are not met for Experiment I.

6.1.2.2 Experiment II

To demonstrate the flexibility of TBRR, we changed some of the task requirements from Experiment I to the values shown in Table 6-5. Note that we lower the force requirement in the Z direction and increase the accuracy requirements from Experiment I.

	Max. Speed (m/s; deg/s)	Force (N; N-m)	Accuracy (mm; deg)
X	0.33	0	1
Y	0.33	0	1
Z	0	250	1
Roll	0	0	2
Pitch	0	0	2
Yaw	0	0	2

Table 6-5: Task requirements for Experiment II.

The results for Experiment II are shown in Table 6-6. TBRR1 again indicates TBRR without buffering. TBRR2 here adds a buffer of 0.5 mm to the accuracy requirement.

Method	Joint Limit Violation	Force Violations	Accuracy Violations	Total
PI	***Y***	*165*	*379*	***544***
GO1	***Y***	*0*	*111*	***111***
GO2	***Y***	*66*	*388*	***454***
GO3	**N**	0	325	**325**
GO4	**N**	0	101	**101**
GO5	**N**	0	242	**242**
CB1	**N**	164	385	**549**
CB2	**N**	161	387	**548**
CB3	**N**	167	383	**550**
CB4	**N**	0	220	**220**
TBRR1	**N**	0	125	**125**
TBRR2	**N**	0	67	**67**

Table 6-6: Number of step points in which task requirements are not met for Experiment II.

6.1.2.3 Analysis of Simulation Results

Several conclusions can be drawn from the simulation results presented above. Let's first look at the simulation results of the traditional RRTs used in the simulations. As expected, because it cannot incorporate any additional performance criteria, the PI method does not yield acceptable solutions in terms of satisfying the system constraints and task requirements. For both experiments, the

PI solutions violated the joint limit constraint and did not meet many of the task requirements in a good number of points along the trajectory.

Analyses of the results from GO and CB both bring forward a few surprises and strongly demonstrate the problem with traditional RRTs with performance criteria. The purpose of the critical boundaries concept is to utilize redundancy more efficiently by limiting the number of performance criteria to be optimized. During normal operation in which no constraint is critical, redundancy is used to optimize operational goal-based criteria. Only when constraint violations are imminent will the associated constraint-based criteria be included. If the critical boundaries concept works as well as it is intended, we should see better results from CB than GO with the same set of criteria and weights. Inspections of the results from both experiments, however, show that this is not always the case. The numberings of the sets of weights in the experiments for GO and CB were intended to match each other (e.g. GO1 and CB1 used the same set of weights) for easy comparison. In both experiments, the joint limit constraint was violated when running GO1 and GO2 but not in CB1 and CB2. GO3 and GO4 outperformed CB3 and CB4, respectively, when it comes to task requirement satisfaction. GO5, in which only JRA and MOT criteria were used, surprisingly showed respectable performance (it yielded better results than almost all CB cases, except for CB4). These unexpected outcomes can be attributed to the *couplings* among criteria. Optimization of either JRA or MOT or both obviously have a profound (and positive, in this case) impact on the task requirement satisfaction performance.

Having presented the discussion on how traditional RRTs fare against one another, now we wish to illustrate the effectiveness of TBRR. As seen in Tables Table 6-4 and Table 6-6, TBRR even without buffering, performed admirably at satisfying the task requirements. It outperformed traditional RRTs, except only for one case (TBRR1 vs GO4 in Experiment II). With buffering employed, TBRR yielded superior results to any other method in this comparative study.

We can clearly see the advantages of TBRR over traditional RRTs. Not only does it better satisfy the task requirements, but it also is able to adapt more effectively. For traditional RRTs, in order to best satisfy the task requirements, the user may need to establish a new combination of performance criteria when the task requirements change. This process is not trivial and requires in-depth experience and is further complicated due to couplings among various criteria. With TBRR, when the task requirements change, the user just needs to supply the TBRR scheme with a new set of task requirements and optionally a set of buffers. In real-world complex tasks, since the task requirements can change rapidly, TBRR can save a lot of deployment effort.

6.1.3 D&D Demonstration

We have seen how TBRR outperforms traditional RRTs in the comparative study presented in the previous demonstration. Although the task performed by the FT10 robot in that demonstration was demanding in the sense that it required a large amount of force and a high degree of accuracy, it only involved limited movement of the robot's EEF (relative to its workspace magnitude) and was not a sufficient representative of real tasks. As such, we establish the following requirements for the tasks in this demonstration:

- Tasks should challenge the constraints of the robot. For instance, some tasks should require the EEF to work near the workspace limits to challenge the robot's joint travel limits and boundary singularities. Some should require fast motion of the EEF to challenge the robot's joint speed limits. Some should require the robot to apply a decent amount of EEF force to challenge the robot's joint torque limits.
- Tasks should be representative of real tasks such as the Decommissioning and Dismantlement (D&D) tasks performed by the Pit Viper robot at Pacific

Northwest National Laboratory (PNNL) and the Dual Arm Telerobot at Oak Ridge National Laboratory (ORNL) [March *et al.*, 2004].

- Similar tasks should be performed by manipulators of different geometries (preferably with different degrees of freedom) to illustrate the generality of TBRR.

In order to meet the above requirements, we propose to test the TBRR scheme on simulations involving a pipe cutting operation, which is part of preparation of a reactor for closure by removing radioactive piping. The pipe cutting task will be performed by two redundant spatial robots. One is the conceptual FT10 robot used in the previous demonstration. With the generous 4 degrees of redundancy, the FT10 robot is an ideal platform to display the impact of redundancy on task performance improvement over a 6-DOF manipulator. However, the FT10 robot is just a conceptual robot that may never be built. Thus, we will also test TBRR on the PA107 robot.

The following parameter values were used in TBRR for all D&D task simulations unless otherwise stated. The JRA criterion with the critical value of 0.2 and MOT with 0.05 were used to avoid joint travel limits and singularities, respectively. The efficiency criterion used in TBRR was the kinetic energy minimization (Eq. (3.54)). The perturbation size $\Delta\theta$ and the momentum term α (see Eq. (3.11)) used in the simulations were 0.2 degree and 0.8, respectively. The simple search pattern on all joints (10 of them for FT10 and 7 for PA107) were used. These translate to totals of 21 options for FT10 and 15 options for PA107.

Pipe cutting is an essential component in D&D activities and is a very demanding task due partly to a large amount of interaction force involved in the cutting process. Several pipes are present and they are oriented in various directions so the robot must cope with multiple orientations while cutting pipes. Naturally, pipes are long and thus some cut locations are close to the robot and some are far away. For some far locations, the robot must almost make a full

stretch to make a cut. Not only does this bring the robot close to boundary singularities but also decreases the robot's ability to apply force in some directions. In order to improve the task performance, the following analysis can be used to determine the task requirement values[22].

- Metal cutting operations require large EEF force in the cutting direction and large EEF moment around the axis perpendicular to the cutting plane.
- The robot should move slowly while making cuts and quickly between cuts.
- Cutting is a robotic operation that requires precise and compliant motion.

6.1.3.1 The FT10 Robot

The FT10 robot was instructed to make a total of six cuts, two in each of the three pipes. Snapshots of the pipe cutting operation by the FT10 robot are shown in Figure 6-6. These pipes were placed such that the furthest cut was almost at the extreme of the FT10's workspace. Figure 6-5 shows the frame assignment used to specify the force and accuracy requirements in the pipe cutting task, which are shown in Table 6-7. Note that these coordinate axes are not the same as those for the robot base. The maximum linear and angular velocities are 0.9 m/s and 154 deg/s, respectively.

---

[22] Note, however, that this analysis is rudimentary. The sole purpose is to obtain reasonable task requirement values in order to evaluate TBRR. Thorough task/tool modeling and analysis that should give more precise task specifications are currently underway by in another research effort at RRG.

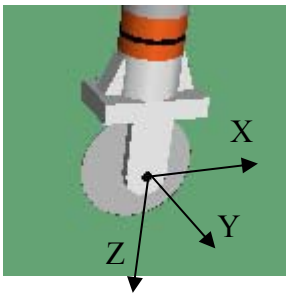
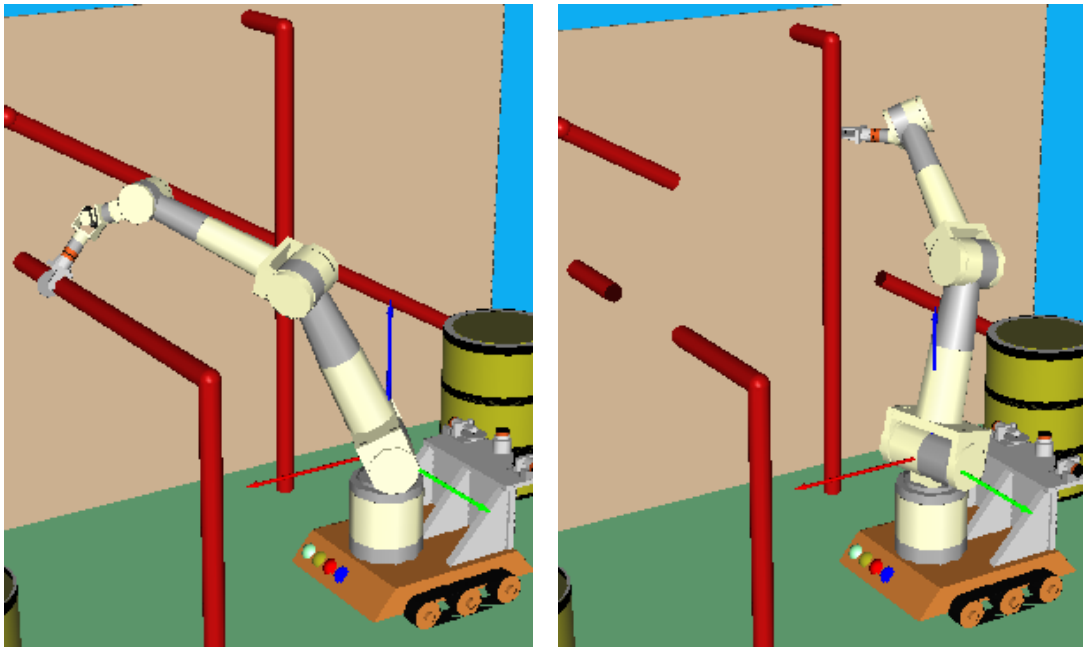


Figure 6-5: Frame assignment for the cutting task.

	Force/Moment (N/N-m)	Accuracy (mm/degree)
X	60	5
Y	0	2
Z	0	2
RX	0	0.3
RY	10	3
RZ	0	0.3

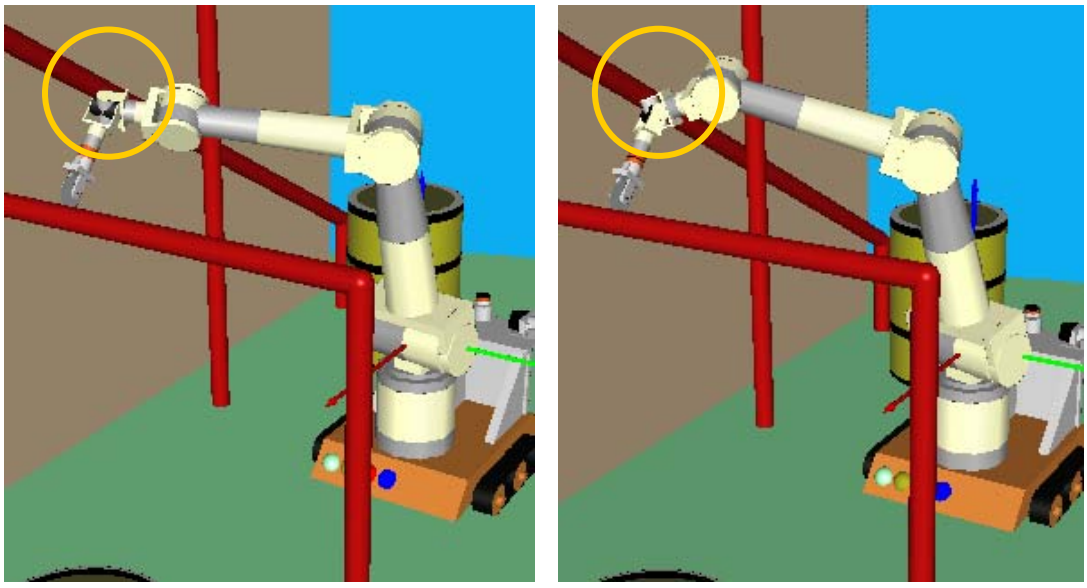
Table 6-7: Force and accuracy requirements for the cutting task by FT10.



(a) Horizontal pipe cutting **(b) Vertical pipe cutting**

Figure 6-6: Pipe cutting by FT10.

The cutting task operation contains 2778 step points with the 0.02 second between each point. The operation with the default parameter values, however, failed at the step point 507 due to a violation of the travel limit on joint 9 (the circle in Figure 6-7(a) highlight the joint 9 violation). It should be noted that joint 9 has a very limited travel range (from 35 to 145 degrees). There are several ways to get around this problem, two of which are presented here. First, we can raise the critical value for JRA (equivalent to increasing the buffer) from 0.2. The tradeoff of this approach is that the bigger buffer size for JRA means that less redundancy is available for satisfying other constraints and task requirements. Second, we can use a different search pattern that generates more than 21 options so that TBRR has a bigger search space. Of course, there is a speed-optimality tradeoff and thus generating more options means a slower computation rate. Both of these approaches were tried and they both succeeded in avoiding the joint limits. Compare pictures (a) and (b) in Figure 6-7, we can see how the robot configures itself to avoid the joint 9's limit violation.



(a) Joint limit violation **(b) Joint limit was avoided**

Figure 6-7: Joint limit violation in the pipe cutting task.

The results of the two solutions are given in Table 6-8. The numbers in the last two columns represent the numbers of step points (out of 2778) in which the requirement (force or accuracy) was not satisfied. As expected, raising the JRA critical value retains fast computation speed but suffers as far as satisfying task requirements. In contrast, increasing the number of generated options yields better task performance but causes the calculation rate to drop.

Change	Computation speed	No. of Force violations	No. of Accuracy violations
JRA critical value from 0.2 to 0.4	130 Hz	155	1068
No. of options from 21 to 65[23]	45 Hz	46	819

Table 6-8: Comparison of two solutions for joint limit violation.

Simulations using traditional RRTs were also carried out for a quick comparison. For the pseudoinverse, joint limit violations expectedly occurred. For the Generate Options method with fixed weights, the following criteria were included: JRA, MOT, VTR, and FTR. Many weight combinations were tried; all but one failed to avoid the limits. The only successful weight combination was one with the weight for JRA criterion was 1 and the weights for the others were 0 (i.e. we tried to optimize JRA only).

6.1.3.2 The PA107 Robot

Table 6-9 shows the force and accuracy requirements for the cutting operation by the PA107 robot. Note that the accuracy requirements are stricter than those for the FT10 robot. This is because we assumed that both of these robots' encoders have the same resolutions and since PA107's reach is about half of

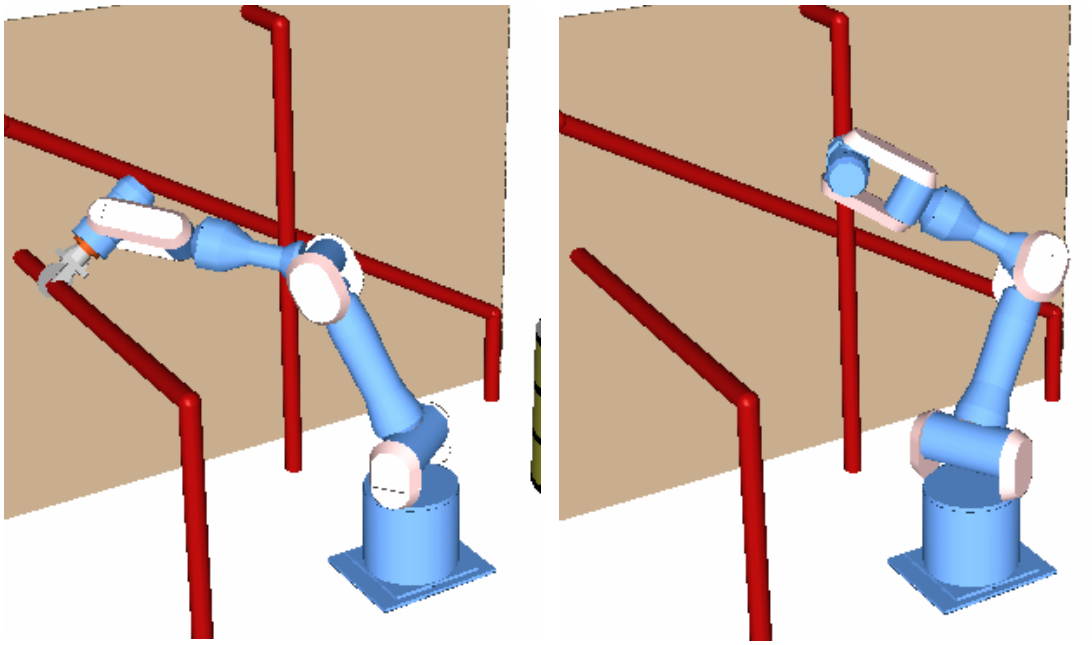
[23] The number of options at 65 was a result of using the factorial search pattern on 6 perturbed joints (i.e. the first 4 joints were not perturbed in creating the null space).

FT10's, it can afford to travel more precisely. The maximum linear and angular velocities are 0.37 m/s and 158 deg/s, respectively.

	Force/Moment (N/N-m)	Accuracy (mm/degree)
X	60	5
Y	0	1
Z	0	1
RX	0	0.2
RY	10	2
RZ	0	0.2

Table 6-9: Force and accuracy requirements for PA107.

Figure 6-8 show snapshots of the PA107 robot performing the pipe cutting task. The overall structure of these pipes' locations and orientations are exactly the same as those used in the FT10 demonstration. However, the pipe sizes and distances were adjusted proportionally to the size of the PA107 robot. Again, the cutting task operation contains 2778 step points with the 0.02 second between each point. Unlike the FT10 simulation, the operation with the default parameters did not violate any system constraint at any time. Also, the velocity and force requirements were satisfied at all times while the accuracy requirement was not met for a total of 249 (out of 2778) step points.



(a) Horizontal pipe cutting **(b) Vertical pipe cutting**

Figure 6-8: Pipe cutting by PA107.

6.1.4 PUMA 760 Demonstration

In this section, we will demonstrate the applicability of TBRR on a commercially available 6-DOF manipulator PUMA 760. In most cases, the PUMA robot is fully constrained. However, when the task at hand requires the output with fewer than 6 DOFs, the PUMA becomes a redundant manipulator and can be utilized as such. A recent development in OSCAR version 2.0 has allowed an easy switch for the inverse kinematics calculations among fully-constrained, under-constrained, and over-constrained configurations of the same robot. The user can *activate* or *inactivate* any of the output directions of the robot's EEF. The OSCAR implementation of TBRR thus takes advantage of this development.

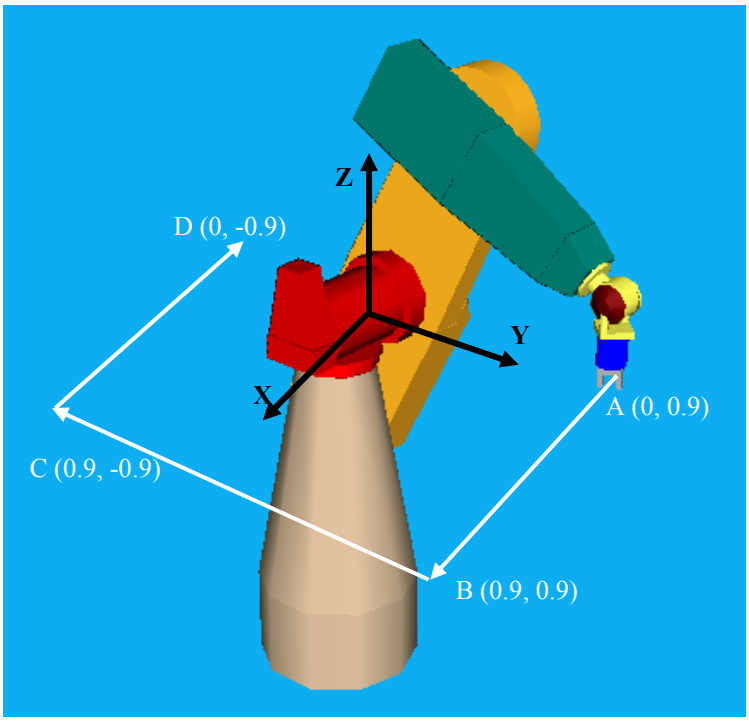


Figure 6-9: PUMA robot performing a tracking task.

Figure 6-9 shows the PUMA robot and a prescribed trajectory. On the X-Y plane, the path consists of three sections, each of which is a trapezoid in velocity. The total number of step points is 1400 in this case. Note that the arm must be almost fully extended to reach the corners (points B and C). The requirements for this task are listed in Table 6-10. In this experiment, we will run the PUMA robot as a fully-constrained robot and as a redundant robot. For the fully-constrained case, the robot is to move its EEF along the prescribed path while maintaining its orientation. As a redundant robot, it will also follow the same path but without concern about its orientation (all three orientation outputs are inactive). The results are shown in Table 6-11. Again, the numbers in the table represent the numbers of step points (out of 1400) in which a specific task requirement is not satisfied. As expected, when the PUMA robot is allowed to act as a redundant robot, it can utilize redundancy to better satisfy the task requirements as seen from the difference in the totals (1095 vs 761).

	Max. Speed (m/s)	Force (N)	Accuracy (mm)
X	0.34	0	0.5
Y	0.45	0	0.5
Z	0	115	0.5

Table 6-10: Task requirements for PUMA experiment.

Configuration	Force Violations	Accuracy Violations	Total
Fully-constrained	212	883	**1095**
Redundant	15	761	**761**

Table 6-11: Results for PUMA experiment.

6.2 Learning Demonstrations

Before we present simulation results of the learning algorithm for buffer sizes that was proposed in Section 5.2.2, let's examine the effects of buffer sizes on the task performance. We learned from the FT10 robot's D&D demonstration in Section 6.1.3.1 that increasing the critical value of JRA criterion from 0.2 to 0.4 helped the FT10 successfully avoid the joint limits when performing the pipe cutting operation. The results in Table 6-12 illustrate the significant impacts of buffer sizes on the task performance. Four different cases from trial and error[24] are presented here. The performance index entries in the final row are calculated using Eq. (5.6) with the weight for constraint violations being 100. Comparing cases A & B, we can see that the buffer sizes of JRA are the same but case B did not violate the joint limit constraint while case A did. This indicates couplings between the joint limit constraint and either force or accuracy requirement or both. Comparing cases B & D reveals a surprising result. With the same buffer sizes for force and accuracy, one would expect that decreasing the buffer size for JRA should improve

[24] We ran many more cases but show only these 4 cases because of their relevance.

the task performance because the robot would use fewer resources to avoid the joint limit constraint. The simulation results contradict this intuition. Case C is the best hand-tuned set of buffer sizes that we found from trial and error.

		A	B	C	D
Buffer Sizes	**JRA**	0.2	0.2	0.2	0.1
	Force (N)	0	5	3	5
	Moment (N-m)	0	0	0	0
	Trans. Error (mm)	0	0.3	0.2	0.3
	Rot. Error (rad)	0	0	0	0
Task Performance	**Joint Constraint Violations**	76	0	0	0
	Force Violations	0	43	50	107
	Accuracy Violations	692	140	109	117
	Performance Index	***8292***	***183***	***159***	***224***

Table 6-12: Effects of buffer sizes on task performance.

Next we applied the learning algorithm that we presented in Section 5.2.2 to find a set of buffer sizes that minimize the performance index. In the simulated training, we chose t to be 10 policies per iteration. In calculating ε for each parameter, the decaying factor η in Eq. (5.7) was set at 0.5 and $\varepsilon_{\max}$ and $\varepsilon_{\min}$ values are given in Table 6-13. For speed, the opportunistic heuristic strategy was employed when selecting the new state for the next iteration. Figure 6-10 shows the performance of the chosen policy after each iteration. The training was stopped after 9 iterations and the performance index of 91 was obtained. Compare this to the best of 159 from hand-tuning and it is obvious that the learning process succeeded. The initial parameter values and final parameter values after training are given in Table 6-13.

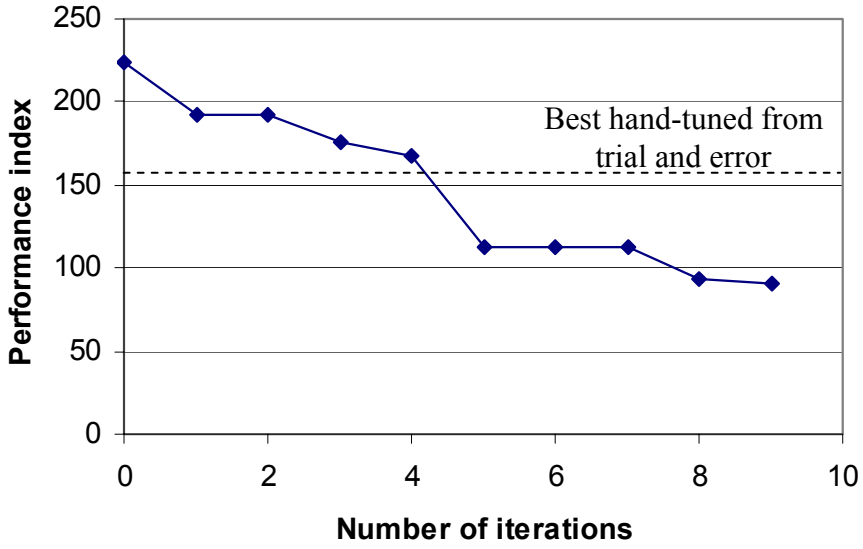


Figure 6-10: Performance index after each iteration during training of FT10 pipe cutting task.

Parameter	Initial	ε_{max}	ε_{min}	Final
JRA	0.1	0.08	0.01	0.284
Force (N)	5	3	0.3	2.52
Moment (N-m)	0	2	0.2	0
Trans. Error (mm)	0.3	0.1	0.01	0.247
Rot. Error (rad)	0	0.0003	0.00003	0

Table 6-13: Initial and final policies as well as ε_{max} and ε_{min} for each parameter used in learning of FT10 pipe cutting task.

In order to illustrate the generality of this learning method, we then applied it to the pipe cutting task performed by the PA107 robot. This task was discussed in Section 6.1.3.2. We used essentially the same learning parameter values (t, η, ε_{max}, ε_{min}, etc.) as in the previous training. Table 6-14 lists the initial and final buffer sizes for this cutting operation. After only 1 iteration, the learning algorithm produced the buffer sizes that yielded the performance index of 20. The learning

stopped after no improvement in performance was detected for 3 consecutive iterations. We did not perform any hand-tuning this time. But the efficacy of the learning algorithm can still be observed when we compare with the performance indices of 207 and 249 obtained from the initial policy and from the simulation in Section 6.1.3.2, respectively.

Parameter	Initial	ε_{max}	ε_{min}	Final
JRA	0.1	0.08	0.01	0.18
Force (N)	5	3	0.3	8
Moment (N-m)	0	2	0.2	0
Trans. Error (mm)	0.3	0.1	0.01	0.2
Rot. Error (rad)	0	0.0003	0.00003	0.0003

Table 6-14: Initial and final policies as well as ε_{max} and ε_{min} for each parameter used in learning of PA107 pipe cutting task.

6.3 Chapter Summary

In the first portion of this chapter, TBRR's effectiveness was illustrated in three demonstrations involving three computer simulated robots. Two of the robots were commercially available. They were a 7-DOF Mitsubishi PA10-7CE and a 6-DOF PUMA 760. The other robot was a conceptual 10-DOF Fault Tolerant robot.

The first demonstration compared TBRR with traditional RRTs on simulations of the 10-DOF robot performing a tracking task and it revealed:

- The major shortcoming of traditional RRTs, i.e. the difficulty of choosing and mixing performance criteria for a given task. This shortcoming stems from the fact that there is no concept of task requirements in any traditional RRT.
- Clear advantages of TBRR over traditional RRTs:
    - Inclusion of task requirements as an integral part in TBRR helps eliminate the above shortcoming.

- Significant improvement on task performance, especially with buffering.

In the second demonstration, the 10-DOF robot and the Mitsubishi robot were instructed to perform more complex and more representative real-world tasks. A pipe cutting operation, which is part of D&D activities, was chosen because of its complex and demanding nature. With a set of default parameter values, the operation by the Mitsubishi robot succeeded while the 10-DOF robot failed due to joint limit constraint violations. This was partly due to a limited travel range on joint 9 of the robot. Two approaches to correcting the failure were suggested. One was to raise the critical value of JRA and the other was to increase the number of generated options. Both fixes were successful and their pros and cons were discussed. This demonstration raised an important issue in how to decide on the values of some subjective parameters in order to maximize the system performance. This issue will be a subject of discussion for the next chapter.

In the final demonstration with the simulations on the PUMA robot, we demonstrated the versatility of the current OSCAR implementation of the TBRR algorithm that allows TBRR to be applied to traditionally non-redundant robots when fewer than 6 DOFs are required for the output. The application of TBRR on non-redundant robots is as direct as setting a few flags on the output directions stating that they are active or inactive. The rest of the process is just the same as that of traditional redundant robots. In addition, we showed that when the PUMA robot acts as a redundant robot, it can perform better as far as task requirements satisfaction is concerned.

The second part of this chapter first demonstrated that buffer sizes had a significant impact on task performance. We showed that changes in buffer sizes could dramatically alter the outcome of the TBRR scheme. In addition, the following two circumstances that make trial and error of parameter assignment difficult were illustrated in the simulations:

- Couplings between system constraints and task requirements when buffer sizes for force and accuracy requirements were increased but the joint limit avoidance was improved.
- Counterintuitive results when the buffer size for JRA was reduced, presumably requiring fewer resources to avoid joint limits, but task performance worsened.

After the effects of buffer sizes had been demonstrated, we presented the results of the parameter learning algorithm proposed in Chapter 5. The purpose was to determine an optimal set of buffer sizes that minimize the number of step points in which system constraints and/or task requirements were violated. In our pipe cutting demonstrations with the FT10 and PA107 robots, the learning algorithm, within a few iterations, was able to find sets of buffer sizes that greatly outperformed those found by hand-tuning. This learning process can be used to possibly replace a time-consuming and tedious chore of trial and error or to enhance the hand-tuned parameters.

CHAPTER 7

FORCE CONTROL IN THE TBRR FRAMEWORK

Chapter 5 introduced the Task-Based Redundancy Resolution (TBRR) method. Generally speaking, TBRR and other RRTs work strictly in the kinematics domain (although they can use dynamics-based criteria). They solve the inverse kinematics problem and send the joint motion commands to the robot controller. The concept of control is usually not figured into the equation of any RRT. All RRTs assume that the robot controller does its job and the joint commands are accurately followed. This assumption is fairly reasonable as far as position control is concerned. However, when force control is required to complete the task, this approach is insufficient.

In this chapter, we will discuss how to integrate force control within the TBRR framework. It must be emphasized that *it is not our goal to develop a new force control technique but only to demonstrate the feasibility of integrating force control with TBRR, thus showing that TBRR can be applied to constrained tasks as well as free-motion tasks*.

7.1 Review of Force Control Methods

This section reviews the existing force control methods found in the literature and focuses on the methods tailored for redundant robots. The purpose of this section is to lay the groundwork on the force control concept and evaluate each control method to determine which one is suitable for a seamless integration with TBRR.

Control of redundant robots is more challenging than that of non-redundant ones because "… the dynamic behavior of the entire redundant system cannot be represented by a dynamic model in coordinates only of the end-effector

configuration." [Khatib, 1987] Khatib was among the first to discuss compliance control of redundant manipulators. Since, for redundant robots, the generalized joint torque vector required to generate a task-space force vector is not unique, Khatib [1990] proposed the following solution for the joint torque vector.

$$\tau = J^T F + (I - J^T J_I^{+T})\tau_0 \tag{7.1}$$

where F is the desired end-effector force vector, τ_0 an arbitrary generalized joint torque vector, J_I^+ the inertia-weighted pseudoinverse (or a generalized inverse of Jacobian corresponding to the solution that minimizes the manipulator instantaneous kinetic energy) given by

$$J_I^+ = M^{-1} J^T \Lambda \tag{7.2}$$

where $\Lambda = \left(JM^{-1}J^T\right)^{-1} \in \mathcal{R}^{m\times m}$ is called the pseudo-inertia matrix. Eq. (7.1) can be viewed as being equivalent to Eq. (3.5) but in the dynamic (force) domain instead of the kinematic (position) domain. The second term of Eq. (7.1) also represents the homogeneous torque and can be used to stabilize the null space motion or optimize a performance index. J_I^+ is said to be dynamically consistent because the null space joint torques associated with it does not produce any acceleration at the end-effector, i.e. $J^T M^{-1}(I - J^T J_I^{+T})\tau_0 = 0$.

Force control methods can generally be divided into two main categories – hybrid position/force control and impedance control. The following sections discuss these two approaches in detail.

7.1.1 Hybrid Position/Force Control

The hybrid position/force control method or hybrid control for short [Raibert and Craig, 1981] is based on the conjecture that when the end-effector is in contact with the environment, the Cartesian space of the end-effector coordinates can be naturally decomposed into a 'position subspace' and a 'force subspace'

[Mason, 1981]. The position/force control problem is then separated into two problems; one tracks a position trajectory in the position subspace and the second tracks a force trajectory in the force subspace.

Peng and Adachi [1993] applied the configuration control [Seraji, 1989] to the resolved acceleration based hybrid control [Khatib, 1987] to obtain an extended hybrid control for redundant manipulators.

$$\tau = MJ_z^{-1}(S_z\ddot{z}^* - \dot{J}_z\dot{q}) + V + G - J^T(I_m - S)f^* \tag{7.3}$$

where S is the $m \times m$ selection matrix specifying the position-controlled directions; I_m is the $m \times m$ identity matrix; (I_m – S) specifies the force-controlled directions. S_z is the $n \times n$ extended selection matrix defined as

$$S_z = \begin{bmatrix} S & 0 \\ 0 & I_r \end{bmatrix}$$

where I_r is the $r \times r$ identity matrix with r being the degree of redundancy. The extended task space Jacobian J_z is

$$J_z(q) = \begin{bmatrix} J(q) \\ J_y(q) \end{bmatrix} \tag{7.4}$$

where $J_y(q) = \partial y / \partial q$ is the $r \times n$ additional Jacobian matrix defined for the r additional kinematic constraints $y = p(q)$. $\ddot{z}^*$ is the extended task space command vector defined as

$$\ddot{z}^* = \ddot{z}_d + K_{zD}(\dot{z}_d - \dot{z}) + K_{zP}(z_d - z) \tag{7.5}$$

where K_{zD} and K_{zP} are respectively the derivative and proportional control gains and $z = \begin{bmatrix} x^T & y^T \end{bmatrix}^T$ is the extended task space vector.

The force-controlled subspace is obtained from the integral control plus a feedforward term as

$$f^* = f_d + K_{fI}\int(f_d - f)dt \tag{7.6}$$

where K_{fI} is a diagonal and positive definite gain matrix.

The extended hybrid control was tested on a 3-DOF planar robot performing a position-tracking task in one direction and a force-tracking task in the other. The transmissibility measure was chosen as the objective function. No attempt, however, was made to extend this to redundant spatial robots.

The subject of explicit force control (which is used in the force-controlled direction of the hybrid approach) has been thoroughly investigated both analytically and experimentally by Volpe and Khosla [1993]. They evaluated a wide variety of force control strategies including proportional with feedforward, proportional-integral (PI), integral, proportional-derivative (PD), filtered PD, and second-order low pass filtering. The conclusion is that the integral gain force control provides the best performance for force trajectory tracking.

The hybrid position/force control is intuitive and simple to implement. However, there are many flaws in this control strategy. First, some tasks such as grinding or cutting cannot be decomposed into position and force subspaces because large amounts of friction along the cutting surface create a constraint that is neither a force or a motion constraint. In other words, these tasks require controlling of force and position along the same direction at the same time. The second problem concerns with decomposing the task space into position and force subspaces. It is implicitly assumed from the selection matrix that the position- and force-controlled directions are *orthogonal complements*. However, as Duffy [1990] pointed out,

> *Modern hybrid control theory based on the definition of orthogonality ... is completely devoid of meaning whether from the point of view of practicability or geometry. For, when we use such a theory to determine whether two given instantaneous motions are "orthogonal," we encounter* (1) *dimensional inconsistency,* (2) *dependence on the choice of units used, and* (3) *dependence on the choice of the origin of coordinates.*

Last but not least, task modeling errors, which are inevitable, could lead to physical damage to the environment, the robot, or both during the task execution.

Consider a task of following a position trajectory in the X-direction while applying a constant force in the Y-direction on a planar surface in Figure 7-1. If the surface is misaligned, the position controller could generate an excessive amount of force in the X-direction as the EEF moves up the slope.

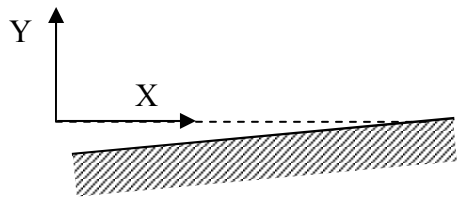


Figure 7-1: Hybrid control with modeling errors.

7.1.2 Impedance Control

The concept of impedance control was first introduced by Hogan [1985]. Unlike the hybrid position/force control, it attempts to regulate the dynamic behavior of the end-effector when in contact, i.e. to maintain a desired dynamic relationship between the end-effector's position and the contact force. Then a mathematical controller is designed such that the desired dynamic behavior can be achieved for a given robot. By selecting proper desired impedance parameters, one can try to control both position and force in the same direction. Let the desired dynamic relationship be represented by the second-order linear equation

$$M_d(\ddot{x}_d - \ddot{x}) + B_d(\dot{x}_d - \dot{x}) + K_d(x_d - x) = f \tag{7.7}$$

where M_d, B_d, K_d are positive-definite matrices representing respectively the desired or target mass, damping, and stiffness; x_d and x are the desired and actual EEF position; and f is the interaction force. For free-space motion ($f = 0$), x approaches x_d. One major drawback of impedance control is that it is impossible to follow a desired force trajectory in any direction. This problem is alleviated somewhat by introducing the desired interaction force into Eq. (7.7). This method, called *Generalized Impedance Control* (GIC), was proposed by Lee and Lee [1991] and later altered by Chan and Liaw [1996].

$$M_d(\ddot{x}_d - \ddot{x}) + B_d(\dot{x}_d - \dot{x}) + K_d(x_d - x) = -(f_d - f) \tag{7.8}$$

Newman and Dohring [1991] introduced the augmented impedance control for redundant robots. This technique combines a *configuration control* approach [Seraji, 1989] with impedance control. It basically augments the Jacobian and the stiffness and damping matrices and uses them in a simplified impedance control law where the inverse of the desired inertia matrix is chosen as $M_d^{-1} = JM^{-1}J^T$. As in the configuration control approach, it is necessary to define a self-motion function (whose dimension is equal to the degree of redundancy) to specify the desired null motion. Although the self-motion function given in the example in the paper is specific to a particular robot and does not have any performance-related physical meaning, this method does provide a means to control the self motion in redundant robots.

Oh *et al*. [1998] proposed a method called an extended impedance control. While the concept of the extended impedance control is similar to that of the augmented impedance control by Newman and Dohring [1991], they are very different in their formulation. The extended impedance control is based on the weighted decomposition of the joint space. As a result, the control joint torques can be implicitly separated into the particular joint torque and null joint torque. Depending on the choice of the weighting matrix, two types of controller can be realized. They are *Kinetically Decomposed Impedance Controller (KDIC)* and *Inertially Decoupled Impedance Controller (IDIC)*. IDIC is of more interest since it decouples the task space and the null motion space inertially. Optimization of a performance index is achieved by using the gradient projection method to specify the null motion.

Nemec and Zlajpah [1999] proposed an impedance control law for redundant robots by introducing the *null torque* term.

$$\begin{aligned}\tau &= J^T[\Lambda(\ddot{x}_d + B_d\dot{e} + K_d e - K_f(f_d - f) - \dot{J}\dot{q}) + f] + V + G + \tau_0 \\ \tau_0 &= M(I - J_I^+ J)(\ddot{q}_{Nd} + (J_I^+\dot{J} + \dot{J}_I^+ J)\dot{q} + K_N e_N + M^{-1}Ve_N)\end{aligned} \tag{7.9}$$

where $\ddot{q}_{Nd}$ is the desired null space acceleration, $e_N = \dot{q}_{Nd} - (I - J^+J)\dot{q}$, $\Lambda = (JM^{-1}J^T)^{-1}$ the operational space kinetic energy matrix, $J_I^+ = M^{-1}J^T\Lambda$ the inertia-weighted pseudoinverse, K_N a diagonal matrix with positive terms.

The null torque ensures the tracking of the null space motion. The desired null space motion can be specified by the gradient projection method such that a performance index $h(q)$ can be optimized.

$$\dot{q}_{Nd} = k(I - J_I^+ J)M^{-1}\nabla h(q) \tag{7.10}$$

where k is a positive real number if the performance index is to be maximized and negative if it is to be minimized. The next derivative $\ddot{q}_{Nd}$ can then be specified by differentiating (7.10) with respect to time. For many complex criteria, it is very difficult to obtain $\dot{q}_{Nd}$ and $\ddot{q}_{Nd}$ in analytical forms. Numerical differentiation is often used and sufficient in most cases. Note here that k determines the rate of convergence of $h(q)$ and a large value of K_N signifies fast tracking performance of the null space velocity.

Pholsiri *et al.* [2003] applied a minimal parameterization of the null space [Oh *et al*., 1998] to Nemec and Zlajpah's impedance control and proposed the *Extended Generalized Impedance Control* (EGIC). The control torque is of the same form as the first line of Eq. (7.9) but the null torque term is

$$\tau_0 = MN\left(\ddot{x}_{Nd} + K_N e_N - \dot{J}_N\dot{q}\right). \tag{7.11}$$

$N \in \mathcal{R}^{n\times r}$ is a full column rank matrix that is made up of a minimal set of linearly independent vectors in the null space of the Jacobian. N can be calculated in many ways, one of which is given by Chen and Walker [1993]. $\ddot{x}_{Nd} \in \mathcal{R}^r$ is the desired null space acceleration. $K_N \in \mathcal{R}^{r\times r}$ is a control gain and defined as a diagonal

matrix with positive terms. $e_N \equiv \dot{x}_{Nd} - J_N \dot{q}$ is the null space velocity tracking error. $J_N \equiv \left(N^T M N\right)^{-1} N^T M$ is the null space Jacobian. Again, the desired null space motion can be specified by the gradient projection method such that a performance index $h(q)$ can be optimized.

$$\begin{aligned} \dot{x}_{Nd} &= k J_N M^{-1} \nabla h(q) \\ &= k \left(N^T M N\right)^{-1} N^T \nabla h(q) \end{aligned} \tag{7.12}$$

The next derivative $\ddot{x}_{Nd}$ can then be specified by differentiating Eq. (7.12) with respect to time. EGIC was successfully tested in computer simulations of a spatial 10-DOF robot.

Even with GIC, force still cannot be explicitly controlled. In GIC, the desired force is only a suggestion and it does not guarantee that the actual force will converge to the desired value. The force tracking accuracy also depends on the desired position, the impedance parameters, and the environment. Consider one-dimensional static force control in a capacitive environment in which the interaction force can be modeled as $f = k_e(x - x_e)$ where k_e is the environment stiffness and x_e is the environment location. Assuming that $x_d = x_e$ (i.e. we want to command the EEF to contact the environment), it is easy to derive the force tracking steady state error from (7.8) as

$$\frac{f}{f_d} = \frac{k_e}{k_e + k_d} \tag{7.13}$$

where k_d is the element of K_d in the direction of interest. For a very stiff surface and low value of the target stiffness $(k_e \gg k_d)$, the actual force approaches the desired value. However, as the target stiffness increases, the force tracking error also increases.

Volpe and Khosla [1995] compared the second-order impedance control and proportional gain explicit force control with feedforward reference force

analytically and experimentally and found them to be essentially equivalent. This led them to "question the value of impedance control as a unified controller for motion through, and constrained interaction with, the environment" due to the following reasons.

- It has been shown that proportional gain explicit force control (and equivalently impedance control) is inferior to integral gain control as far as force trajectory tracking.
- Impedance control always incorporates force feedback information even when the robot is in free-space motion. Erroneous force readings that may result from sensor noise or inertial loading at the EEF may impair the position tracking performance.
- Impedance control parameters that are stable during constrained and free-space motion may cause oscillation or instability during contact transition.

7.1.3 Other Approaches

Over the last decade, researchers have attempted to find new force control schemes that overcome the drawbacks and take advantage of the strengths of hybrid control and/or impedance control. Hybrid Impedance Control (HIC) was proposed by Anderson and Spong [1987] and a robust HIC by Liu and Goldenberg [1991]. As in the hybrid control, the task space in HIC is also decomposed into the position and force subspaces. Unlike hybrid control, impedance control is used in the position controlled directions to provide an acceptable dynamic behavior in the case of unplanned contacts with the environment. Shadpey *et al.* [1995] then proposed an Augmented HIC (AHIC) for redundant manipulators by combining the robust HIC with the singularity-robust configuration control [Seraji and Colbaugh, 1990]. HIC was applied to both the end-effector space and the additional task space, allowing "the control of additional tasks by a force control scheme."

Chiaverini and Sciavicco [1993] developed the parallel approach, which is a force/position control technique that "offers some robustness with respect to the uncertainties affecting the knowledge of the environment." The parallel control is similar to the hybrid control in the sense that the torque commands from the two independent systems – a position controller and a force controller – are added. Whereas the hybrid control uses a task model (via the selection matrix) to handle inconsistent requirements between position and force control actions, the parallel control is designed such that the force control action dominates the position control action. This strategy is intended to avoid excessive forces due to task modeling errors. In order to achieve the force dominance, a PI force control loop is used in parallel to a PD position loop.

$$\tilde{f}_f = K_f e_f + K_i \int_0^t e_f d\tau \tag{7.14}$$

where $e_f = f_d - f$ is the force error; K_f and K_i are the force loop proportional and integral gain matrices, respectively. A resolved acceleration position controller is used for position control action.

$$\tilde{f}_p = M_d \ddot{x}_d + K_v \dot{e}_p + K_p e_p \tag{7.15}$$

where $e_p = x_d - x$ is the position error; M_d is the positive definite desired inertia matrix that must be diagonal to ensure dynamic decoupling in the task space; K_v and K_p are the position loop derivative and proportional gain matrices, respectively. The total torque command to the robot is simply the sum of the two commands.

$$\tilde{f} = \tilde{f}_p + \tilde{f}_f \tag{7.16}$$

The closed-loop dynamics (see Chiaverini and Sciavicco [1993] for the derivation) can be written as

$$M_d \ddot{e}_p + K_v \dot{e}_p + K_p e_p + K_f e_f + K_i \int_0^t e_f d\tau = 0 . \tag{7.17}$$

It can be seen from Eq. (7.17) that e_f dominates e_p because of the integral term. Interestingly, the closed-loop dynamics obtained via the parallel control can be considered as an extension of either the impedance control (with an added force control capability) or the explicit force control (with a position control consideration). The parallel control was successfully tested on a 6-DOF industrial robot, but with the wrist being locked and only linear force and motion are of interest.

7.2 Basic Implementations of Force Control

Two basic implementations of most force control schemes are torque-based[25] (or direct) and position-based. In torque-based force control, the commanded torque is computed directly from the control laws such as those in Eq. (7.3) or (7.9). Apparently, this requires that the robot be capable of being torque-controlled at the joint level. Most industrial robots, however, are usually position-controlled via Independent Joint Control (IJC), i.e. closed PID loop on the actuator, and no access to the low-level controller is given. In addition, the complete dynamic model of the manipulator, which is needed for many force control laws including those in Eq. (7.3) or (7.9), is either not available or of poor quality [Ferretti *et al.*, 2000]. Therefore, torque-based force control cannot be implemented on most industrial robots. The implementation of direct force control is straightforward and therefore no further discussion is necessary.

---

[25] We could use the term 'force-based' in place of 'torque-based' but force-based force control would sound redundant. Also, since most modern industrial robots use revolute joints, the term 'torque-based' is probably more appropriate.

7.2.1 Position-Based Force Control (PBFC)

Position-based, a.k.a. inner position loop, force control do not require direct access to actuator torques and therefore can be implemented on most commercial robots. The outer force loop provides a reference position to the inner position loop through a *modifier* as shown in Figure 7-2. The modifier uses force feedback to modify the desired position trajectory (x_d) and then passes on the commanded EEF position (x_c) to the inverse kinematics solver. The commanded joint position (q_c) is then solved and sent to the robot controllers. The robot joint controllers are commonly PID controllers specially tuned for the particular robot. They are usually very robust and have good position tracking capability. This is partly due to the fact that most industrial robots' actuators employ high gear ratio transmission which greatly reduces the coupling among actuators and disturbances from external forces.

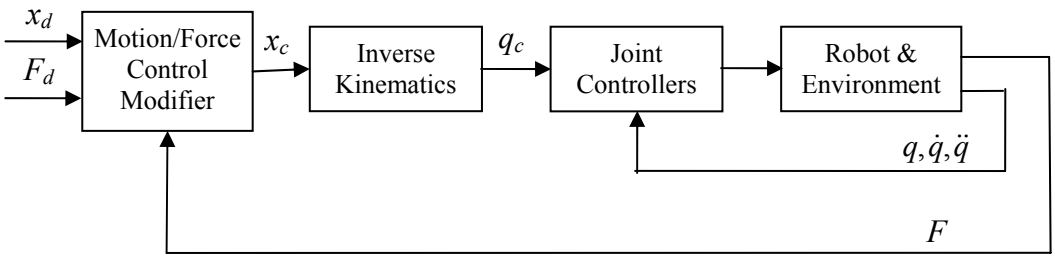


Figure 7-2: Position-based force control.

Maples and Becker [1986] pointed out that using the inner position loop force control could significantly increase the disturbance rejection capability. It would also allow us to apply the split-rate technique on the servo controller (faster servo rate for the inner position loop and slower servo rate for the outer force loop), thus reducing the computational burden because the outer force loop computation was usually more complicated. Their PBFC implementation was successfully

tested on an AdeptOne robot on an assembly task of parts having 0.001 inch clearance.

Volpe and Khosla [1993] showed that, from a control theory point of view, PBFC only differs from the torque-based force control in that PBFC adds an additional stiffness term to the plant. This additional stiffness term could potentially have a destabilizing effect on the system. Furthermore, Heinrichs and Sepehri [1999] determined that finite positioning accuracy of the joint controllers could lead to recurring oscillation of the contact force in a prolonged static force control application in PBFC unless the target stiffness doubles the stiffness of the combined environment and force sensor.

Perhaps the most common type of PBFC is *Position-Based Impedance Control* (PBIC). In PBIC, the modifier realizes the following generalized impedance relationship.

$$M_d(\ddot{x}_d - \ddot{x}_c) + B_d(\dot{x}_d - \dot{x}_c) + K_d(x_d - x_c) = -K_f(F_d - F) \tag{7.18}$$

If the robot can perfectly track the commanded EEF position, i.e. $x = x_c$, then (7.18) becomes (7.8) and the target impedance relationship is realized. The inverse kinematics then determines the commanded joint trajectory, which is in turn fed into the robot joint controllers.

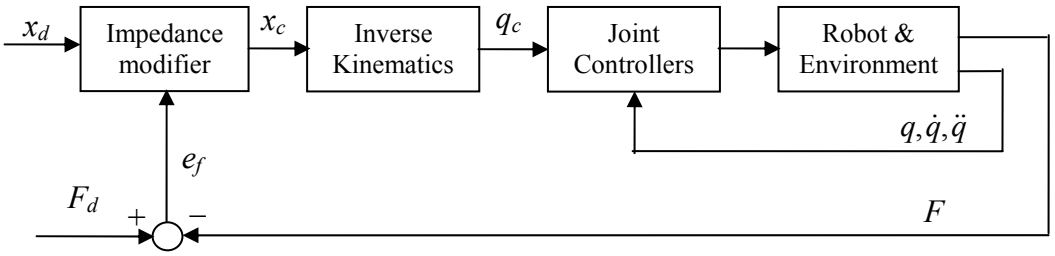


Figure 7-3: Position-Based Impedance Controller.

The particular PBIC method illustrated in Figure 7-3 is called the *straightforward* method because it uses only force feedback but no state feedback to shape the commanded EEF position. It assumes that the position tracking of the

joint controllers is perfect and consequently does not attempt to correct any error. Pelletier and Doyon [1994] proposed three methods that use position, velocity, and acceleration feedback in the impedance filter to construct the commanded EEF position trajectory. Surdilovic and Kirchhof [1996] considered the linearized model of the robot and the joint controller properties. Ferretti *et al.* [2000] then expanded Surdilovic and Kirchhof's method to include joints with elastic transmission.

Reynolds *et al.* [1993] applied PBIC to an 8-DOF gantry robot for a robotic circular saw cutting of packages containing radioactive waste at the Savannah River Site. PBIC was used to control the three translational axes. It made necessary trade-off between perfect position tracking and excess force generation. Although the task completion time was longer than if the trajectory was completed using traditional control methods, PBIC successfully executed the cutting task without an occurrence of saw binding.

Table 7-1 summarizes the pros and cons of torque-based and position-based force control implementations.

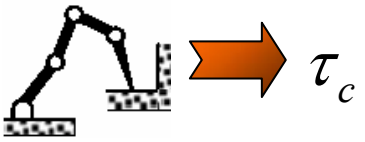
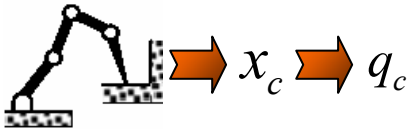
	Torque-Based	**Position-Based**
Concept		
Pros	• No inverse kinematics required • Faster dynamic response	• Uses joint's position controller • No dynamic model needed • Disturbance rejection
Cons	• Requires ability to torque-control joints • Requires accurate dynamic model or advanced adaptive control	• Requires inverse kinematics solution • Excellent joint controller is required for high-performance force control

Table 7-1: Comparison of torque-based and position-based force control.

7.3 Integration of TBRR with Force Control

Again, it must be emphasized that the goal of this chapter is to demonstrate the feasibility of integrating force control with TBRR. The selected force control method may not necessarily be the best as far as the control performance is concerned.

7.3.1 TBRR and Direct Force Control

Our first attempt to integrate TBRR with force control was naturally with direct (or torque-based) force control methods. As seen from the review in the previous section, two of the most common ways to incorporate redundancy into force control are the extended Jacobian method and the gradient projection method. The key to a successful integration may lie in how to make TBRR compatible with one of these methods. For the extended Jacobian method (see Eqs. (7.3) - (7.5) for example), the null space torque command is implicitly specified in the extended Jacobian and the extended task space vector. On the other hand, in Eq. (7.9), the null torque term that ensures the tracking of the null space motion is specified explicitly and the null space motion specification is then obtained via the gradient projection method shown in Eq. (7.10). Clearly, if we are to choose between the two methods, the gradient projection should be simpler for TBRR integration because the null space torque specification is explicit. Following this logic, we decided to use Eq. (7.9) in our first attempt of TBRR integration with force control.

As discussed in the previous chapter, TBRR is a *position-level inverse kinematics* method that incorporates constraints and task requirements while resolving the redundancy. If we are able to obtain the null space motion specification ($\dot{q}_{Nd}$ and $\ddot{q}_{Nd}$) from the TBRR solution, then we can use them directly in Eq. (7.9). Here, instead of specifying the desired null space motion with the gradient projection method as in Eq. (7.10), we first solved the inverse kinematics

problem using TBRR, numerically differentiated the solution, and then projected it into the null space.

$$\dot{q}_{Nd} = (I - J_I^+ J)\left(\frac{q_{new} - q_{current}}{\Delta t}\right) \tag{7.19}$$

where q_{new} is the solution obtained from TBRR; $q_{current}$ is the current configuration; Δt is the time step; and $(I - J_I^+ J)$ is the null space projector.

However, the preliminary results obtained from computer simulations of the Planar4R and FT10 robots did not show good promise of this approach. In fact, the system even showed some instability in many of the runs.

7.3.2 TBRR and PBFC

The failed attempt of integrating TBRR with the direct force control method above forced us to take a step back and look at not only the control itself but also how it is implemented. Because TBRR is a *position-level inverse kinematics* method, the position-based force control implementation is probably more appropriate than the direct implementation. Thus, we decided to pursue PBFC as the force control implementation of choice to be integrated with TBRR. Note that PBFC is in itself not a force control method but only an implementation. One benefit of PBFC that we did not address in Section 7.2.1 is that it can be used to implement explicit force control, impedance control, or any other force control method. If the underlying force control method that we choose does not perform as well as we expected, then we can easily change it to another.

Here, we chose the parallel approach (see Section 7.1.3) due to its abilities to track both force and motion trajectories (a main drawback of the impedance control) and to cope with unplanned contact forces (a major weakness of the hybrid position/force control). PBFC allows a straightforward integration of TBRR and force control, with the TBRR module just replacing the inverse kinematics module

in Figure 7-3. The additional inputs to the TBRR module are a set of task requirements and the actual states of the robot.

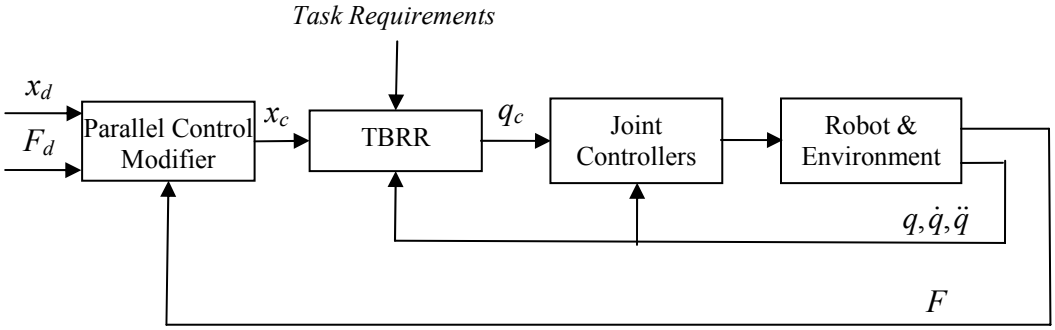


Figure 7-4: Integration of TBRR with parallel control implemented by PBFC.

The parallel control modifier realizes the following relationship.

$$M_d(\ddot{x}_d - \ddot{x}_c) + K_v(\dot{x}_d - \dot{x}_c) + K_p(x_d - x_c) + K_f(F_d - F) + K_i \int_0^t (x_d - x_c) d\tau = 0 \quad (7.20)$$

If the robot can perfectly track the commanded EEF position, i.e. $x = x_c$, then Eq. (7.20) becomes Eq. (7.17) and the closed-loop dynamics of the parallel control is realized.

7.4 OSCAR Implementation

This section presents the software implementation of the integrated force control system discussed in Section 7.3.2 in OSCAR. From Figure 7-4, the parallel control modifier, TBRR, and the joint controller need to be implemented. Also, to test the integrated force control system in a computer simulated environment, we also need a dynamic simulator to simulate the robot behavior. Since TBRR implementation has already been covered in Chapter 5, this section is devoted to discussing the implementation of the other components.

7.4.1 Control Library

OSCAR provides tools for calculating kinematics, dynamics, performance criteria, etc. One omission from OSCAR's vast array of libraries is the robot control domain. Since we need to implement the control method described in the previous section in OSCAR, we have to create a control library as a prerequisite. The control framework should comply with the three important OSCAR requirements: generality, extensibility, and reusability. We should be able to use it to control any serial manipulator. It should provide methods for basic control laws such as PID, computed-torque control, etc. It should be able to accommodate both position and force control. Also, it should be expandable so that new and improved control laws can be easily added without destroying the whole structure of the library and applications. Care must be taken to ensure that the control library allows for generality, extensibility, and reusability as much as possible. Note, however, that because software implementation is only a minor part of this research effort, the control framework implemented here will only be preliminary and are subject to further improvement by future researchers. In addition, only the control components necessary for this research will be implemented.

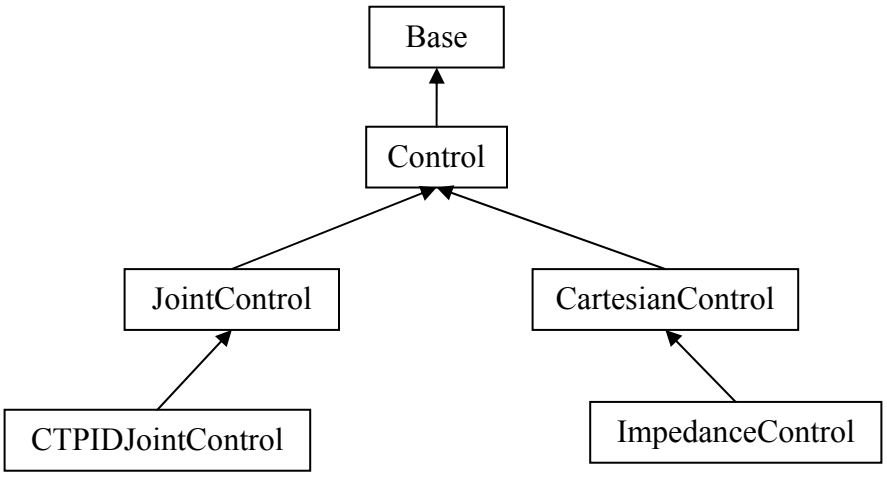


Figure 7-5: Class Hierarchy for the Control Domain.

Control is implemented as an abstract data type as it only provides an interface common to all control methods. It contains only two virtual methods: *SetParameters(parameters)* for setting control parameters and *GetControlTorques(controlTorques)* for outputting the control torques. Their implementations are left to the derived classes. To realize both Cartesian and joint space control, ***CartesianControl*** and ***JointControl*** classes, respectively, have been developed. While ***CartesianControl*** contains the methods to set the actual and desired EEF trajectory in *SetCartesianStates(* $x, \dot{x}, x_d, \dot{x}_d, \ddot{x}_d$ *)* as well as the actual joint states in *SetJointStates(* $q, \dot{q}$ *)*, ***JointControl*** has only the method to set the actual and desired joint trajectory in *SetJointStates(* $q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d$ *)*. One specific joint controller, namely computed-torque PID controller (discussed in Section 2.3.1.2), has been implemented as ***CTPIDJointControl*** class. Because impedance control deals with the dynamic relationship between the EEF and the environment, it is clear that ***ImpedanceControl*** should be derived from ***CartesianControl***. Since force information is required in any impedance control method, *SetForces(* f, f_d *)* method is added to ***ImpedanceControl*** class.

More advanced control laws such as robust or adaptive control can be easily added under this software structure. However, they have not been implemented as they are beyond the scope of this work.

7.4.2 Parallel Control Modifier

For implementation purposes, the force/motion control filter in Eq. (7.20) must be discretized. Because the desired closed-loop dynamics of the parallel control is decoupled, it is sufficient to consider a one-dimensional case. Here, the command bias can be expressed as

$$e_c(p) = \frac{\left[\left(\frac{2m}{T^2} + \frac{k_v}{T}\right)e_c(p-1) - \frac{m}{T^2}e_c(p-2) - k_f e_f(p) - k_i \varepsilon_f(p)\right]}{\left(\frac{m}{T^2} + \frac{k_v}{T} + k_p\right)} \quad (7.21)$$

where $e_c(p) \equiv x_d(p) - x_c(p)$ is the position bias at time p; $e_f(p) = f_d(p) - f(p)$ is the force error at time p; m, k_v, and k_p are elements of M_d, K_v, and K_p, respectively; k_f is an element of K_f; $\varepsilon_f(p) = \sum_{l=0}^{p} e_f(l)$; and T is the sampling period. The commanded EEF position can then be computed from $x_c(k) = x_d(k) - e_c(k)$.

7.4.3 Dynamic Simulator

In addition to the control library, dynamic simulation is also necessary. The dynamic simulator is composed of forward dynamics and an integrator. The forward dynamics implemented here is calculated using inverse dynamics. The interested reader should refer to Featherstone [1987] for the details of how to use inverse dynamics to calculate forward dynamics. We use the inverse dynamics based on the Newton-Euler method because it is much faster than the Lagrange-Euler method[26]. This implementation is simple, though not as efficient as some other methods. More efficient methods, such as the composite-rigid-body method or the articulated-body method [Featherstone, 1987], would require too many components, especially in spatial arithmetic and dynamics, to be added to OSCAR and that is beyond the scope of this research. An integrator is needed to integrate the accelerations from the forward dynamics to obtain the manipulator's joint velocities and positions. The integrator used in the simulation is the widely-used

---

[26] Both the Newton-Euler and Lagrange-Euler inverse dynamics algorithms are readily available in OSCAR.

4th-order Runge-Kutta method. The class diagram of the dynamic simulator is shown in Figure 7-6.

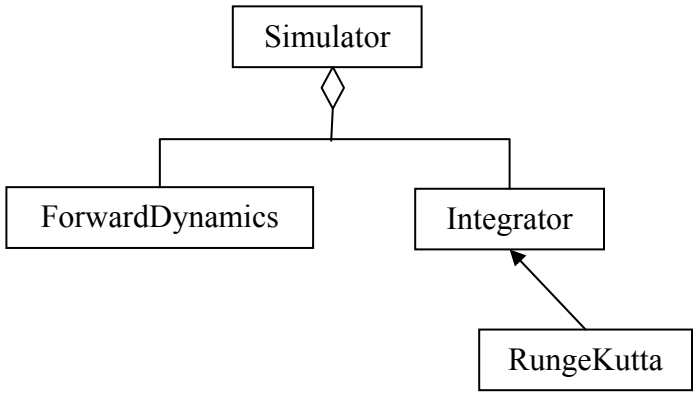


Figure 7-6: Class Diagram for Dynamic Simulator.

7.5 Simulation Results

This section presents simulation results of the integrated system of TBRR and force control method as discussed in Section 7.3. The force control was tested on the FT10 robot. In order to emulate a realistic robot, we also included the dynamics of the actuators, especially the motor inertias and transmission's speed ratios. These properties can be found in Appendix A.

As previously explained, one benefit of PBFC is the possibility of a split-rate controller where the outer force feedback loop is slower than the inner position control loop. In our simulations, the outer force feedback loop has a sampling rate of 33 Hz. The PID computed-torque control (Section 2.3.1.2) is used in the joint controllers with the control rate of 1 kHz. The proportional, derivative, and integral gains on each joint are set to 3600, 120, and 10, respectively. The integration step in the 4th-order Runge-Kutta method for dynamics simulation is also taken at 1 millisecond. A smaller integration step of 0.1 millisecond was also attempted but yielded almost exactly the same results. In all simulations, we use the following control parameters (Eq. (7.20)) unless otherwise stated:

$$M_d = I_6,\ K_v = 400I_6,\ K_p = 400I_6,\ K_f = 0.3I_6, \text{ and } K_i = 0.2I_6 \qquad (7.22)$$

where I_6 is the 6×6 identity matrix.

7.5.1 Experiment I: Motion and Force Tracking

Here the FT10 robot performs a motion and force tracking task. The robot is to track the edge of the circular lid with a radius of 31.75 cm, which lies on a plane parallel to the X-Y plane (see Figure 7-7), while exerting force in the –Z direction. The motion trajectory is planned by a trapezoid at the velocity level. For simplicity, the friction force (of the contact) is not considered and the linear contact force model is assumed, i.e. $f_z = K_e \Delta z$, where the contact stiffness K_e = 20000 N/m. The force trajectory to be tracked is shown in Figure 7-9. The total time of execution is 18 seconds.

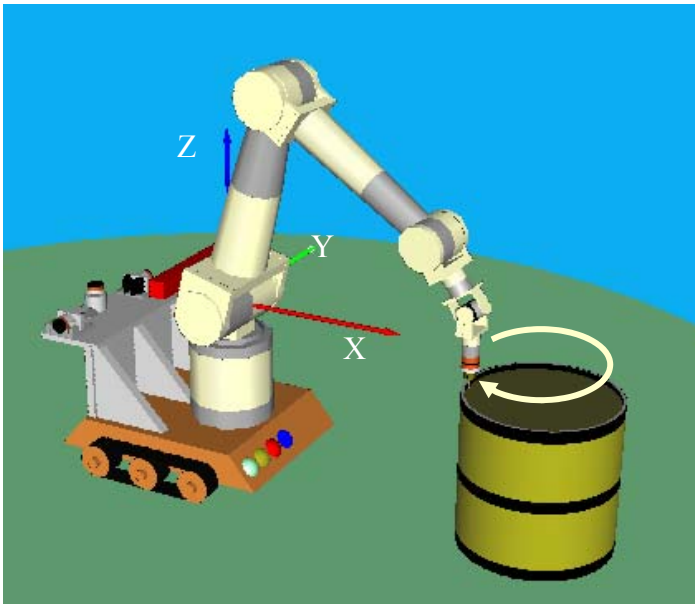
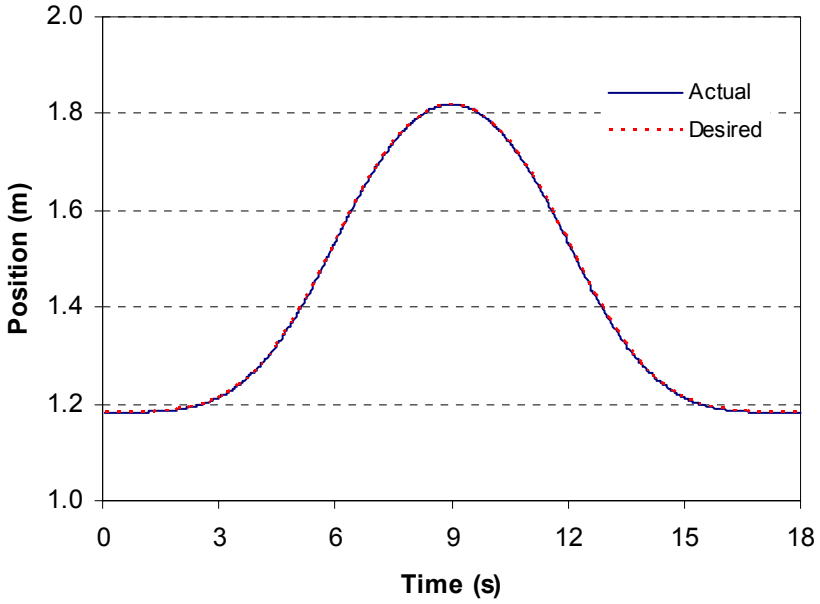
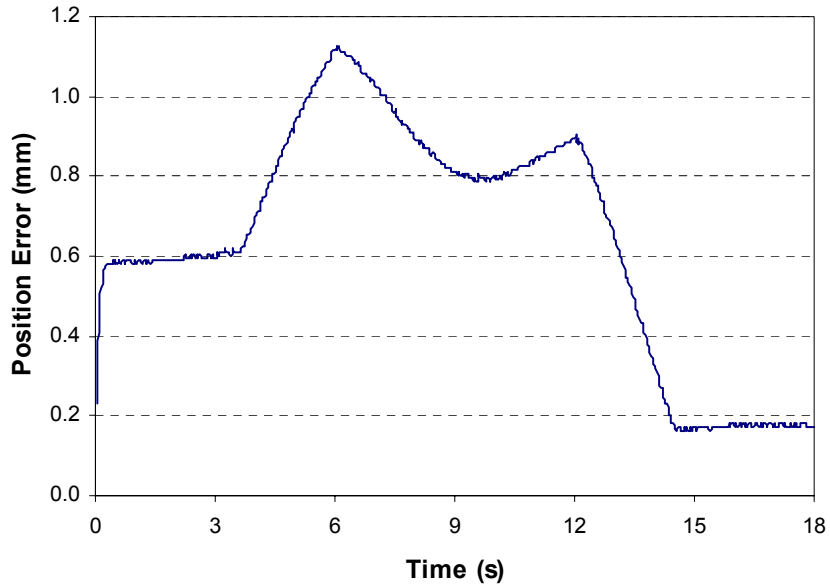


Figure 7-7: FT10 Manipulator performing the tracking task.

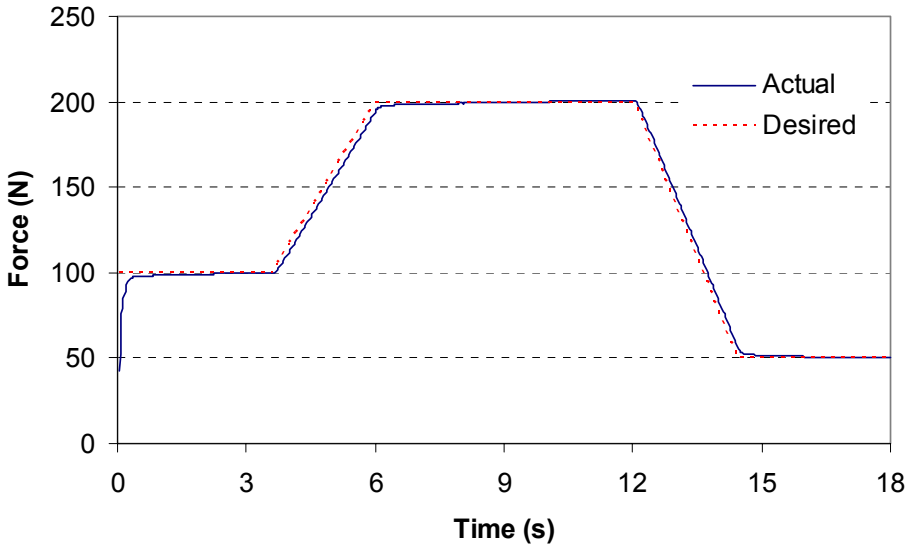


(a) EEF position

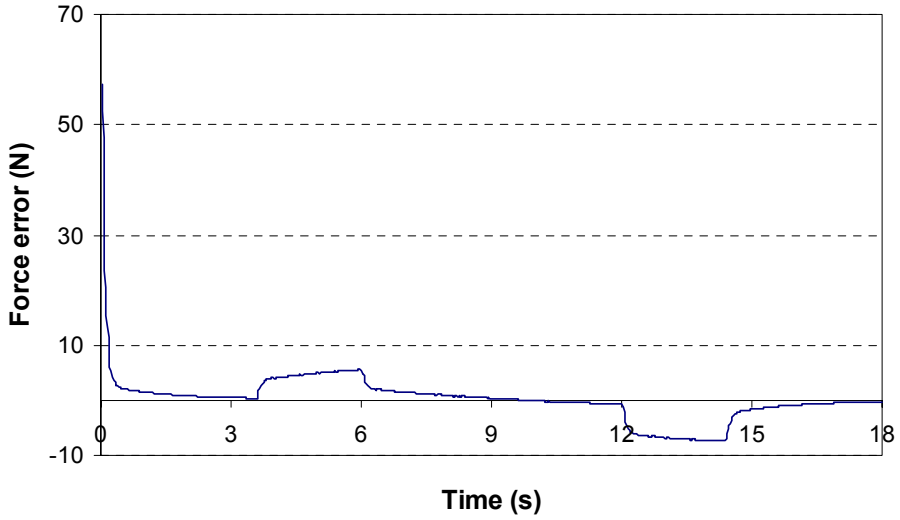


(b) Position error

Figure 7-8: Position tracking in Experiment I.



(a) Contact force



(b) Contact force error

Figure 7-9: Force tracking in Experiment I.

Figures 7-8 and 7-9 show the tracking performance for position and force trajectories, respectively. We can see that the robot followed both the position and force trajectories very well. The maximum position error is less than 1.2 mm. The

force tracking showed no overshoot with a fast rise time for the step input and a slight lag for the ramp input.

7.5.2 Experiment II: Unplanned Contact Force

The task here is interesting in that it consists of both free space motion and contact motion and the contact motion is not planned. Here, the FT10 robot is to follow a circular trajectory on the X-Z plane centered at (x, z) = (1.5 m, 0 m) with a radius of 0.3175 m. However, there is a wall parallel to the Y-Z plane locating at x = 1.7 m (a vertical dotted line in Figure 7-10), obstructing the motion of the robot. Again, we model the wall as a stiff linear spring with K_e = 20000 N/m with no friction. If the robot was purely position-controlled, then the peak force could reach 20000*(1.8175-1.7) = 2350 N. This high amount of force is too excessive and could easily damage the robot. We need the contact force to be significantly lower than this.

Since the planned task is of the free motion type, the desired force (to be fed into the control scheme) is zero in all directions for the whole trajectory. All control parameters are borrowed from the previous experiment. The goal of this experiment is to illustrate how the integrated force control system copes with unplanned contact force.

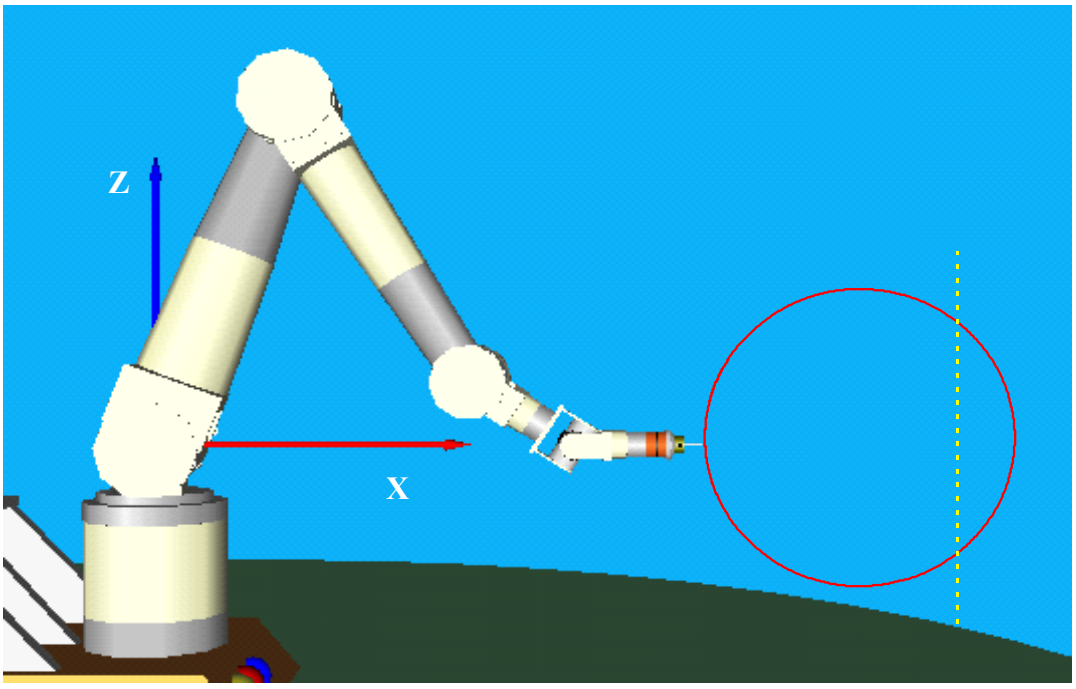


Figure 7-10: Desired trajectory for Experiment II.

First the default control parameters in Eq. (7.22) were used in this experiment with the result shown in Figure 7-11. It is obvious that the actual path deviated significantly from the desired trajectory after the contact with the wall. Upon closer inspection of Eq. (7.20), it was discovered that the situation arose because the integral term $K_i \int_0^t (x_d - x_c) d\tau$ that formed during contact never vanished even after contact, causing the motion error afterwards. Therefore, the integral gains K_i were modified such that they were zeros in free space and assumed the value of 0.2 during contact.

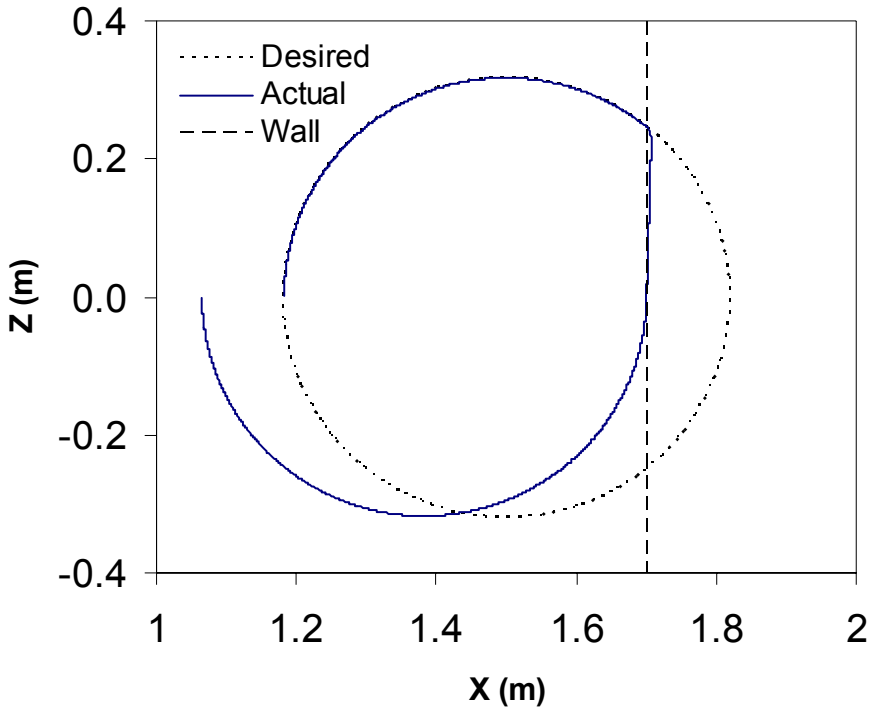
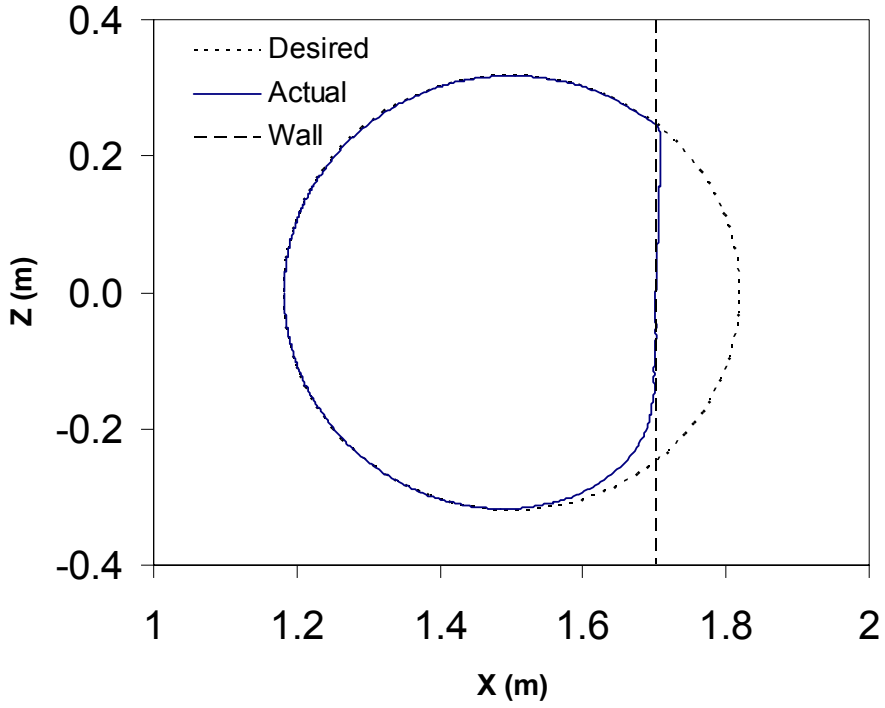
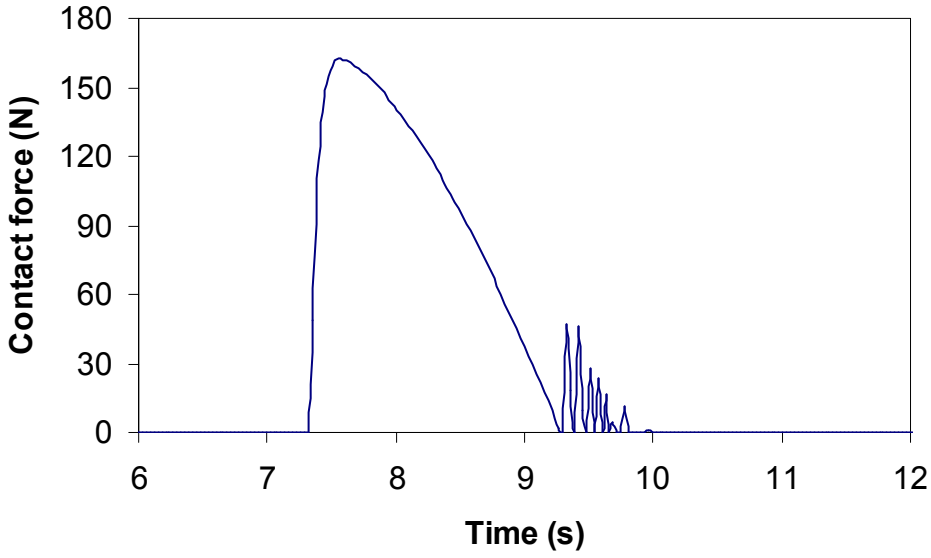


Figure 7-11: Experiment II with default control parameters.

Figure 7-12 shows the result of this experiment with the modified variable gains. The actual path deviated slightly after contact but eventually converged to the desired trajectory. This is a significant improvement compared with the previous run with fixed control gain. Also, the contact force peaked at around 160 N, which is more than 14 times lower than the 2350 N that would have resulted with the pure position control. The force chatter observed at the end stems from the switching of the integral gains between 0 (free space) and 0.2 (contact). Therefore, the integrated force control system did in fact achieve its goal of considerably reducing the unplanned contact force while respectably following the motion trajectory.



(a) Cartesian position



(b) Contact force

Figure 7-12: Experiment II with variable control gains.

7.6 Summary

The purpose of this chapter was to demonstrate a feasibility of integrating force control with TBRR. We first reviewed several existing force control methods and determined that most force control methods for redundant robots used either the extended Jacobian or gradient projection to resolve redundancy. We then tried to integrate TBRR with one of the force control methods that we reviewed by using TBRR to specify the null space motion. However, our preliminary investigation suggested that this approach would not work well as it sometimes caused instability. So we took a step back and looked at the implementations of force control methods in general and found that there were two main implementations, namely torque-based and position-based. The advantages and disadvantages of the Position-Based Force Control (PBFC) were discussed. PBFC was found to be very suitable for TBRR because TBRR solves the inverse kinematics at the position level. The integration of TBRR and PBFC was straightforward. The underlying force control method was the parallel control [Chiaverini and Sciavicco, 1993].

The integrated system of force control with TBRR was tested on the FT10 robot with acceptable results. Two types of experiments were conducted. In the first experiment, the robot successfully tracked both position and force trajectories. In the second, the integrated force control effectively reduced unplanned contact force by more than 93% compared to the pure position control, resulting in a much safer working condition. However, we *do not* claim that the combination of TBRR and the parallel control implemented by PBFC is the best performing force control method for redundant manipulators. We only claim that it has been shown here that TBRR can be used in constrained tasks involving interaction forces as well as free-motion tasks.

Byproducts of this work are a dynamic simulator package added to the existing OSCAR's dynamics library and a working, albeit preliminary, version of a control library added to OSCAR. The dynamic simulator allows the user to

observe the dynamic behavior of any serial manipulator given the input torques and external EEF loads. A major attribute of the control library is that we can effortlessly employ a number of control laws to any serial manipulator without deriving symbolic expressions for such elements as Jacobian or inertia matrix, owing to the generality of OSCAR software libraries. Also, control laws can be swapped out and interchanged without affecting the rest of the application. This can be a significant time saver when developing and testing new control laws.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This chapter contains the Conclusions and Future Work sections. In the Conclusions section, we first reiterate the goal of this research and then summarize the steps that we took in order to achieve that goal, with emphasis on major contributions of this research. Future Work section discusses further developments necessary to advance the research on decision-making and control of robotic systems. Some of these developments are in fact currently underway at UTRRG.

8.1 Conclusions

Redundant robotic systems are versatile yet complex. They possess extra resources, resulting in multiple solutions (choices) for a given set of constraints. The conventional approach to choosing an optimal solution has largely been based on forming a composite performance index from a set of performance criteria. With literally tens of criteria to choose from (many of which are practically cryptic to non-expert users), the user is faced with an undue burden of choosing the right mix of these criteria. Many researchers have tried to alleviate this burden by coming up with *strategies* that dynamically adjust the weight of each criterion. While these strategies have shown promise, almost all of them fail to take into account the very basis of robotic tasks – *task requirements*. The main goal of this research was to solve this very problem that has long hindered wider acceptance of redundant manipulators.

- Develop a decision-making scheme for redundant manipulators that incorporates a concept of *tasks* as an integral part.
- To this end, the following components of the decision-making system must be thoroughly examined:
    - Redundancy Resolution Techniques (RRTs)
    - Performance criteria
    - Task requirements and robot capabilities
- Integrate force control with the decision-making scheme so that it can be applied to both contact and non-contact tasks.
- Facilitate or automate the selection process of system parameters to make the task-based approach more accessible for non-expert users.

Table 8-1: Summary of Research Objectives.

This report presented an integrated framework for *task-based decision making and control* in redundant systems. It must again be emphasized that redundant robots here refer to ones that possess greater DOFs than required to perform a task at hand. These include all 6-DOF commercial robots performing tasks requiring 5 or fewer output dimensions. The framework consists of several essential components that were developed following a comprehensive review and evaluation of existing supporting technologies including manipulator modeling, mathematical tools (linear algebra, vector and matrix norms, etc.), performance criteria, redundancy resolution techniques, motion and force control methods, and optimization methodologies. This section summarizes the works done in this research. Particular emphasis is placed on the primary contributions that this research has provided.

8.1.1 Redundancy Resolution Techniques

Several Redundancy Resolution Techniques (RRTs), which are a core component of the decision making system, were thoroughly reviewed in Section 3.1. Pros and cons of each technique are given in Table 3-1. Upon a careful evaluation, the Generate Options (GO) method developed at UTRRG, which is based on the Direct Search technique, was chosen as an RRT of choice. Although the GO algorithm had many benefits, it possessed some weaknesses in its implementation that need to be addressed as well.

This research continued an ongoing effort of improving the GO scheme. Two major problems with the existing GO were:

- Algorithmic singularities due to locked joints and the use of partial Jacobian matrix.
- Tradeoff between trajectory smoothness and optimality.

Two improvements were suggested in Section 3.1.6 to address these two problems. The first improvement was to use the full Jacobian matrix instead of a partial Jacobian in the inverse solution. This greatly improves the chances of avoiding algorithmic singularities. In addition, the use of full Jacobian also leads to an expansion of the search space and a greater likelihood of finding better solutions.

The tradeoff between trajectory smoothness and optimality in the current implementation of GO was another problem. In order to obtain a better solution, the perturbation size usually needed to be large. A large perturbation size, however, may cause joint 'wiggle' that leads to a non-smooth trajectory. One could introduce joint velocity and acceleration constraints to filter out such options but at the expense of reducing the search space and therefore solution quality.

To cope with this tradeoff problem, we introduced a *momentum* term to the GO method in order to smooth out the solution trajectory. Our premise for adding the momentum term is that the current solution should have a bearing on the next

solution and thus should be present in the formulation in some form or another. The formulation of the improved GO with the momentum term is given in Eq. (3.11). By adding the momentum term, we can use a smaller joint perturbation size to achieve similar optimality with larger perturbation sizes while increasing the smoothness of the solution trajectory. Our simulation in Section 3.1.8 showed that the momentum term could improve the trajectory smoothness by 79% while maintaining the similar optimality of the solution. Table 8-2 summarizes the improvements on the GO method suggested in this research.

- Problems with the existing GO
  - Algorithmic singularities due to locked joints and the use of partial Jacobian matrix.
  - Tradeoff between trajectory smoothness and optimality.
- Improvements to the existing GO
  - Use of full Jacobian
  - Introduction of momentum term
- Use of full Jacobian matrix
  - Greatly reduces, if not eliminates, the occurrence of algorithmic singularities.
  - Expands search space and thus increases likelihood of finding better solution.
  - Provides flexibility of locking *any* combination of joints and thus allows greater freedom of choosing the size of the search space.
  - More computationally intensive than the existing GO because a larger matrix must be inverted.
- Momentum term
  - Allows us to use a small perturbation size to achieve similar optimality of a large perturbation size in the existing GO.
  - Results in a smoother solution trajectory.
  - Simulation showed an almost 79% improvement in smoothness with similar optimality.

Table 8-2: Summary of the improved Generate Options algorithm.

8.1.2 Performance Criteria

Section 3.2 carefully reviewed a large collection of the more useful performance criteria in the literature. Performance criteria are used as a measure of the quality of the solution in RRTs. Criteria are usually optimized to gain

performance benefits. Performance criteria were categorized into two classes: constraint-based criteria and operational goal-based criteria. Constraint-based criteria are associated with system (physical or mathematical) constraints. Their formulations and interpretations are normally straightforward and easily understood even by non-expert users. Operational goal-based criteria, on the other hand, are meant to help the system achieve operational goals. Their relations to tasks, though well intended, are however somewhat vague.

Individual criteria were then discussed in detail. A summary of the criteria reviewed in Section 3.2 along with their interpretations and significances to tasks was given in Table 3-2. Finally, current issues of performance criteria were discussed. These issues are some of the main reasons performance criteria are difficult to use and interpret and in fact gave motivation to this research effort. See Table 8-3 for listing of the described criteria and discussed issues.

- Constraint-based criteria
  - Joint limit avoidance
  - Velocity limit avoidance
  - Peak torque avoidance
  - Obstacle avoidance
  - Mathematical singularity avoidance
- Operational goal-based criteria
  - Dexterity
  - Speed of operation
  - Load carrying capacity
  - Manipulator precision
  - Energy minimization
  - Cyclic or conservative motion
- Issues with performance criteria
  - Scaling
  - Normalization
  - Level of redundancy resolution
  - Task dependency
  - Couplings and conflicts among criteria

Table 8-3: Summary of performance criteria.

8.1.3 Robot Capability Estimations via Vector Expansion Method

Traditionally, robot capabilities are estimated by constructing either velocity/force ellipsoids or polytopes (see Figure 8-1 for example). Ellipsoids are analytically simpler and can be easily calculated in real time. However, they do not give accurate results because they are based on the 2-norms of the joint capability limits vectors. Polytopes, on the other hand, are computed from the ∞-norms of those vectors and thus provide accurate estimations of robot capabilities.

Nonetheless, one must construct a polytope numerically and the numerical process is very time-consuming. For example, Hwang *et al*. [2000] computed a velocity polytope for a 10-DOF in a 3D workspace using an efficient iterative method that was almost 65 times faster than a conventional method, yet it still took 6.64 seconds on an HP 700 system to construct just one polytope. The computational complexity of polytopes renders them impractical for real-time monitoring and control applications in which the cycle rate of 30 Hz is recommended.

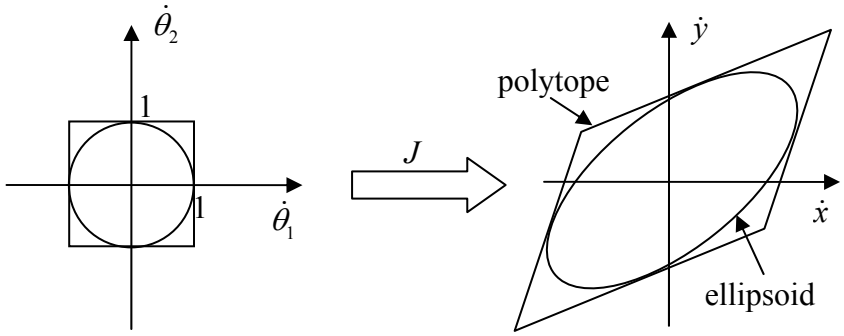


Figure 8-1: A 2-DOF example of ellipsoid and polytope (repeated from Section 4.2.1.1).

To overcome the weaknesses of the ellipsoid and polytope methods, the Vector Expansion (VE) method was developed in Chapter 4 as a tool to accurately and quickly estimate robot capabilities, given the robot's configuration and properties. The VE method is based on the ellipsoid expansion method proposed by Bowling and Khatib [1995] to analyze the isotropic acceleration characteristics of a 6-DOF manipulator.

The VE method works in a similar fashion as the ellipsoid expansion method (Figure 8-2) but, instead of analyzing isotropic properties (which is useful for design), it analyzes the direction of interest. It first maps a vector (the direction of interest) from the EEF space onto the joint space and then expands or contracts the mapped vector until it reaches the bounding box composed of joint capability limits.

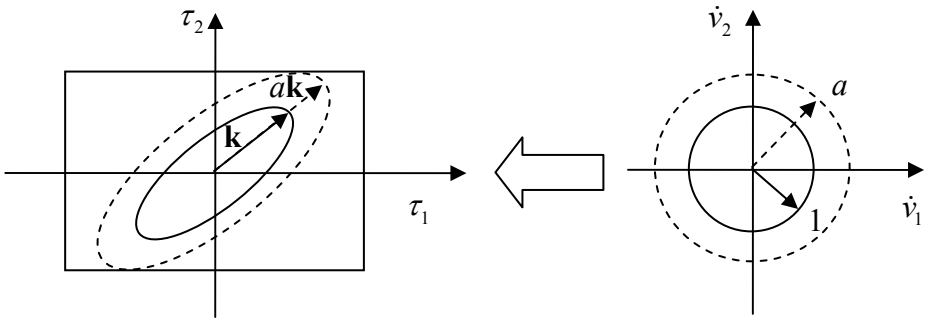


Figure 8-2: Ellipsoid expansion and reverse mapping (Adapted from Bowling and Khatib [1995]).

We have developed and implemented the VE formulations for estimating the following robot capabilities:

- EEF achievable speed – this is the maximum EEF speed given the joint speed limits (Section 4.2.2).
- EEF achievable acceleration – this is the maximum EEF acceleration given the joint torque limits, assuming that the robot is stationary or moving very slowly and there is no external load (Section 4.2.3).
- EEF positional error – the positional error at the EEF given the joint errors (Section 4.2.4).
- EEF static force – the maximum EEF force the robot can apply given the joint torque limits, assuming that the robot is stationary or moving very slowly (Section 4.2.5).

These formulations were used to construct polytopes for a 4-DOF planar robot. The polytopes were then compared against the corresponding ellipsoids. The results were consistent with what we had expected in that ellipsoids yielded conservative estimates compared to polytopes and that the major and minor axes of ellipsoids and polytopes did not necessarily coincide. Furthermore, the discrepancies between the estimates from the ellipsoid and the polytope were large in some direction and negligible in others. Figures 8-3 and 8-4 show examples of

using the VE method to estimate the force capability of a 4-DOF robot and also compare the polytopes obtained via the VE method with the force ellipsoids. The numbers along the edges of the polytopes are the limiting joints. As can be seen, the ellipsoid method yields conservative estimations where some directions are more accurate than others, with inaccuracy of as much as 33% being observed in this case. The impact of gravity on the robot's force capability is also clearly illustrated by comparing the results from these two figures.

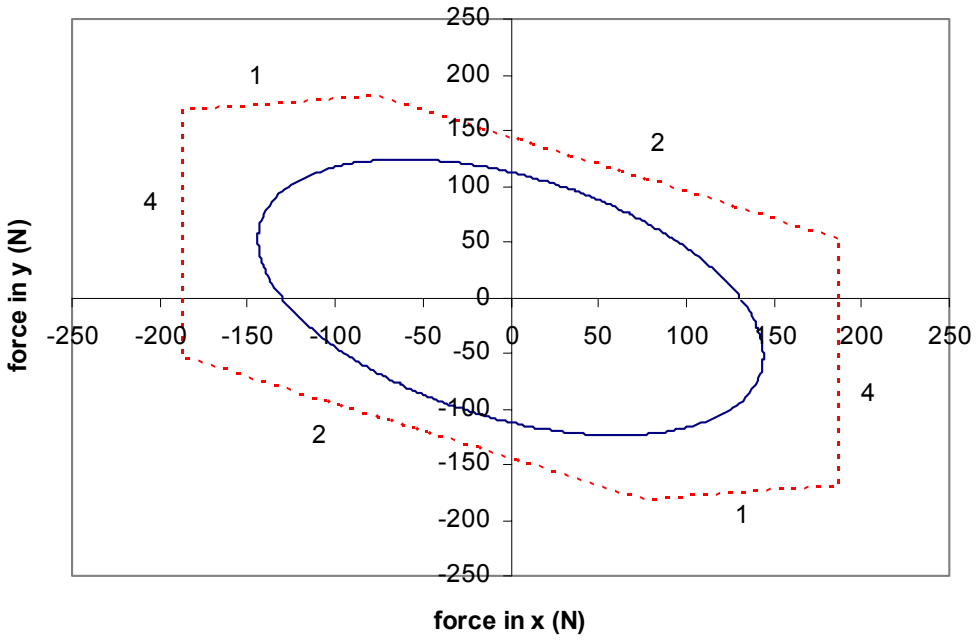


Figure 8-3: Force ellipsoid (solid) and force polytope (dotted) without gravity (repeated from Section 4.2.5.3).

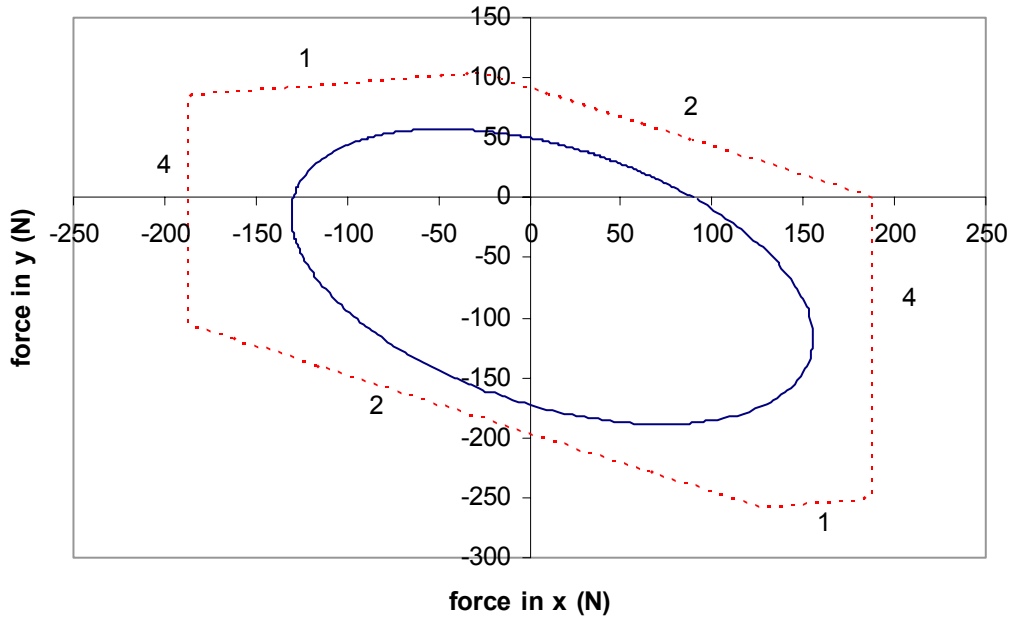


Figure 8-4: Force ellipsoid (solid) and force polytope (dotted) with gravity (repeated from Section 4.2.5.3).

Because the robot capability estimations use the joint capability limits in their calculations directly, they are readily equipped to cope with faults or partial failure of the joints. Joint partial failure is referred to when one or more joints can no longer supply the torques at their full strength. When the system detects joint partial failure and determines the reduced torque capabilities for the failed joints, it can use these new values directly in the VE formulations. The new force and/or acceleration capability based on the reduced joint torques can then be computed in the redundancy resolution process or to create a sense of margin of failure. Another benefit of the VE method over the ellipsoid method is that one can obtain the limiting joint (i.e. the joint whose capability limits the EEF capability) information without extra computation. The limiting joint information is particularly useful for reconfiguring modular manipulators.

The newly developed VE method for quickly and accurately estimating robot capabilities is summarized in Table 8-4.

- Robot capability estimation using the ellipsoid approach
  - Inaccurate due to the use of 2-norm.
  - Closed-form solution thus fast computation.
- Robot capability estimation using the polytope approach
  - Accurate because it uses ∞-norm.
  - Numerical solution thus computationally expensive especially for high-DOF robots.
  - Not suitable for real-time operations.
- The VE approach was developed to overcome weaknesses of the ellipsoid and polytope approaches
  - Based on reverse mapping borrowed from the ellipsoid expansion method.
  - Fast and accurate estimations for capability in a certain direction, thus suitable for operational purposes.
  - Does not yield the whole capability volumes, which is useful in the analysis and design of manipulators although one could construct them from the VE method by sweeping the direction vector to cover the whole workspace.
  - Provides the limiting joint information.
  - Is a great tool for fault tolerance and failure recovery.
- Robot capability estimations using the VE method were formulated and implemented for
  - EEF speed
  - EEF acceleration
  - EEF static force
  - EEF positional error

Table 8-4: Summary of the VE method for estimating robot capabilities.

The following applications can greatly benefit from robot capability estimations:

1. **Task-based decision making**

In task-based decision making, robot capability estimations are used in the redundancy resolution process to determine configurations (in the null space) that satisfy task requirements and eliminate those that do not. Therefore, a solution trajectory obtained in this manner will ultimately meet the specified task requirements (as much as physically possible), enhancing the user-perceived task execution performance. This is what we used robot capability analysis for in this report.

2. **Online and offline path planning**

In online path planning, robot capabilities are continually monitored and measured against task requirement specifications. When one or more robot capabilities diminish and approach the values of associated task requirements, the system can then alert the user or better yet send information to an online path generator so it can adjust the robot path accordingly. Robot capability estimations can also help an offline path planner to generate trajectories that will ensure that task requirements can be met throughout. Note that the use of robot capability analysis to online and offline path planning is equally applicable to both redundant and non-redundant robotic systems. Robot capability analysis coupled with offline task planner can have a significant impact on industrial systems where non-redundant robots are widely accepted.

3. **Design and/or assembly of robotic manipulators**

Robot capability analysis can be an invaluable tool in designing robotic manipulators and/or assembling modular ones from modules (e.g. actuators and links). In the design and/or assembly process of a manipulator, a number of target tasks are specified and robot capability analysis (together with other tools) can then be used in an iterative procedure of determining the required joint capability

(speed, torque, etc.) limits. In fact, an investigation of this subject is underway in a parallel UTRRG development effort.

8.1.4 Task-Based Redundancy Resolution

Chapter 5 started off by pointing out the deficiencies of traditional multi-criteria RRTs that resolved redundancy by optimizing a composite performance objective formed by linearly combining multiple performance criteria. These deficiencies were the main motivations for us to develop Task-Based Redundancy Resolution (TBRR). TBRR together with the VE method is probably the most significant contribution made in this research. To the best of our knowledge, this is the first complete effort to bring the decision making process up to the task level, in which the operator does not need to fully understand the intricacy of redundancy resolution process and ambiguous interpretations of many performance criteria.

TBRR is unique in the sense that it revolves around exploiting redundancy to help the robot perform the task at hand successfully. That means, the extra resources must first be used to help the robot perform the task without violating any constraint and satisfying speed, accuracy, and force requirements. The remaining resources can then be utilized to minimize the energy consumption or other criteria as desired. The flowchart of TBRR is depicted in Figure 8-5. TBRR begins with a set of options that satisfy the EEF constraints. Then, the sequential filtering technique [Eschenbach and Tesar, 1969] is applied to these generated options in the solution selection stage. For each option, the constraint-based criteria are evaluated against the critical values specified by the user. The options that do not satisfy all the constraints are *filtered out* and only the options that do are passed on to the next step. The constraints are given priority because of their utmost importance. Next, the robot capabilities are estimated and then evaluated against the desired values of the task requirements specified by the user. Again, the options that do not satisfy all the task requirements are filtered out. The best configuration can then be chosen

from the remaining options by using one of the criteria for energy minimization or other types of criteria as desired. A concept of *buffering* was introduced as a means to improve the system performance by allowing TBRR to react more promptly when a violation of any constraint is imminent. Buffering in effect increases performance reserves (robot capabilities minus the required performances). A *conflict resolution* method was also presented to mediate when none of the generated options satisfies all constraints and/or task requirements. In essence, the conflict resolution assigns a score to each options corresponding to how much it violates the constraints. Then, the option with the lowest score (least violation) is then chosen. Then a guideline on how to select proper critical values was presented.

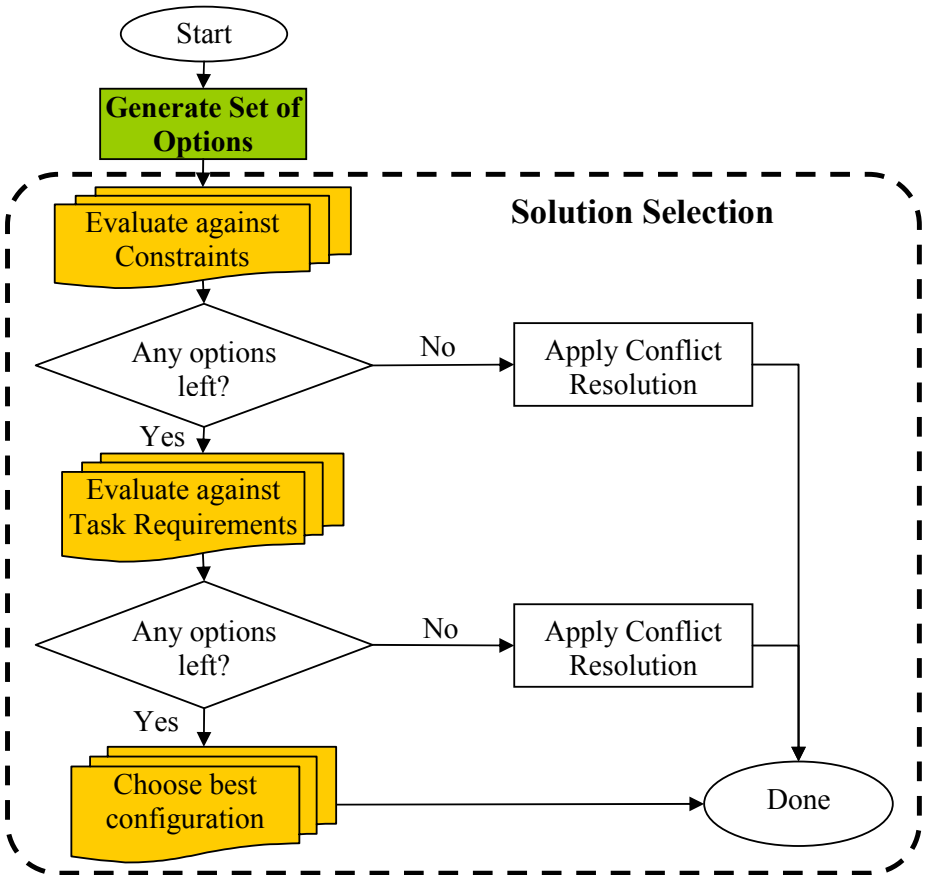


Figure 8-5: TBRR operational flowchart (repeated from Section 5.1.1).

Most traditional RRTs use a weighted sum of multiple performance criteria as the performance index in their optimization stage. It is left to the user to select the set of criteria and assign their relative weights that he feels appropriate for the task at hand. This requires that the user have deep understanding and experience with all performance criteria and how they relate to the task. TBRR, on the other hand, allows a direct integration of task requirement specifications in the decision-making process. As such, TBRR

- greatly simplifies the user's role,
- is more responsive to changes in task requirements, and
- tends to yield better task execution performance.

The following table summarizes the main differences between traditional RRTs and TBRR.

TBRR	Traditional RRTs
• Relies on performance criteria and robot capability estimations • Uses sequential filtering and a single criterion at the end • User supplies critical/desired values for system/task constraints	• Rely on performance criteria alone • Use weighted sum of multiple criteria • User chooses criteria and assigns weight to each criterion

Table 8-5: Comparison between TBRR and traditional RRTs.

TBRR was implemented using the OSCAR framework. In order to gauge TBRR's generality and versatility, in Chapter 6 we conducted three demonstrations on three simulated robots of different geometries. These robots included two commercially available robots in a classic 6-DOF PUMA robot and a new lightweight 7-DOF Mitsubishi PA10-7CE robot (PA107); and a conceptual 10-DOF Fault Tolerant robot (FT10). For the PUMA robot, the goal was to illustrate the applicability of TBRR on a traditionally non-redundant robot.

The first demonstration was a comparative study intended to evaluate the effectiveness of TBRR against traditional RRTs. In this demonstration, the FT10 robot was used to track a circular path while applying a force onto an

environmental surface. The force and accuracy requirements were deliberately set at high levels to highlight the performance of TBRR. This demonstration revealed the difficulty of choosing and mixing performance criteria for a given task, which is a major drawback of traditional RRTs. In addition, it also showed that TBRR outperformed traditional RRTs by 3%-147% when it comes to satisfying task requirements.

The goals of the second demonstration were two fold. First, we wanted to test TBRR in a demanding task that was also representative of real-world tasks. Second, we wished to illustrate versatility by having two kinematically different manipulators perform the same task. Hence, a pipe cutting task was chosen and it was performed by the two redundant PA107 and FT10 robots. With a set of default parameter values, the operation by the PA107 robot succeeded while the FT10 robot failed due to joint limit constraint violations. This was partly due to a limited travel range on joint 9 of the robot. Two approaches to correcting the failure were suggested. One was to raise the critical value of JRA and the other was to increase the number of generated options.

The PUMA robot, which is typically considered non-redundant, was used in the third demonstration. The purpose was to evaluate whether or not 6-DOF manipulators could benefit from TBRR if their tasks did not require all six DOFs. The results showed that when the PUMA robot acts as a redundant robot, it can perform better as far as task requirements satisfaction is concerned.

The development and benefits of TBRR are summarized in Table 8-6 along with the simulation results and observations.

- Development of TBRR
  - TBRR uses the improved GO method along with three steps of sequential filtering.
    - System constraints
    - Task requirements
    - Efficiency or any other criterion as desired
  - Buffering increases performance reserves and thus better prepares the robot for unforeseeable future.
  - Conflict resolution is applied when no option satisfies all system constraints and/or task requirements.
- Benefits of TBRR
  - Simplifies the user's role by replacing the process of selecting and combining multiple performance criteria for a given task with a simpler and more intuitive task of selecting critical values and desired values for system and task constraints.
  - Information about performance reserves is readily available. This information can be invaluable not only in the decision making but in condition-based maintenance and fault tolerance as well.
- Demonstration results and observations
  - The difficulty of selecting and combining performance criteria for a given set of task requirements was demonstrated.
  - TBRR provides 3%-147% improvement over other RRTs and is more responsive to changes in task requirements.
  - Changing buffer sizes and/or perturbation space affects the system performance.
  - TBRR can be applied to traditionally non-redundant robots.

Table 8-6: Summary of TBRR and its demonstrations.

8.1.5 Learning of Subjective Parameters

Like any other RRTs, TBRR possesses several subjective or experience-based parameters. The user generally employs the tedious and time-consuming trial and error approach when tuning the system parameters. To help the user cope better with parameter subjectivity in TBRR, we proposed to use a learning algorithm to determine an optimal set of buffer sizes (critical and desired values) for a given task in Section 5.2.2. By this we mean, a set of buffer sizes that minimize the number of step points in which system constraints or task requirements are violated. Parameter learning automates the search for good values of system parameters. The learning algorithm embedded understanding and logic of TBRR in the form of probability when searching for a new policy (set of buffer sizes). Two heuristic strategies, namely hill-climbing and opportunistic, were used to guide the search.

In our demonstration in Section 6.2, we were able to find good values for buffer sizes using the learning algorithm for pipe cutting operations by the FT10 and PA107 robots. The result for the FT10 robot is depicted by the plot in Figure 8-6. The performance index is defined in Eq. (5.6) and smaller values indicate better performances. We can see that after 5 iterations the learning algorithm yielded the result that was better than the best hand-tuned buffer sizes from trial and error. Table 8-7 summarizes the parameter learning approach developed in this research.

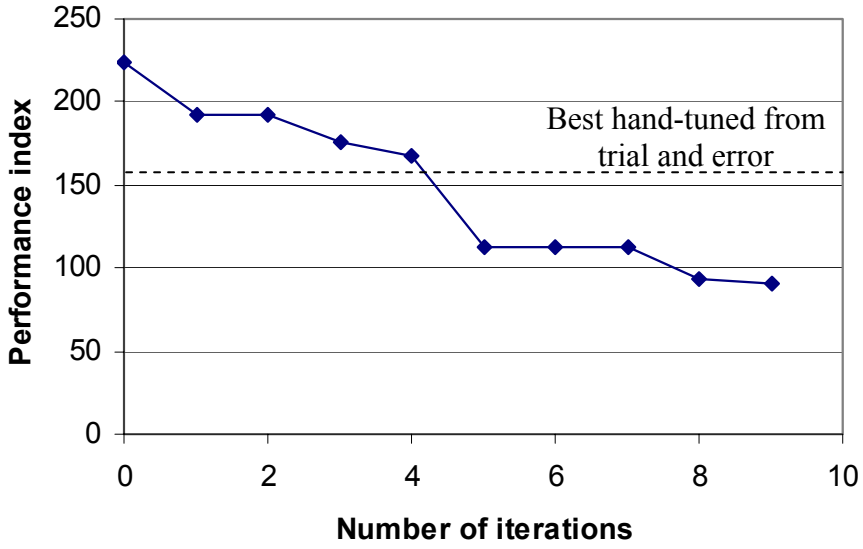


Figure 8-6: Performance index after each iteration during training of FT10 pipe cutting task (repeated from Section 6.2).

- A learning algorithm to automate the search of proper buffer sizes was developed.
  - New trial states were generated *probabilistically* using understanding of TBRR.
  - Variable perturbation was used for faster convergence.
  - Hill-climbing or opportunistic strategy was used to guide the search.
- Simulations show the success of the learning algorithm when compared with hand-tuning.
- The learning approach reduces the user's reliance on trial and error, saving time and frustration.

Table 8-7: Summary of parameter learning.

8.1.6 Force Control in the TBRR framework

One of the goals of this research was to demonstrate the feasibility of integrating force control with TBRR. Most RRTs only concern themselves with kinematics and fail to take into account the dynamics and control issues in their redundancy resolution process. They usually assume that the robot controller does its job and the joint commands are accurately followed. This assumption is fairly reasonable as far as position control is concerned. However, when force control is required to complete the task, this approach is insufficient.

In Chapter 7, we employed the parallel approach [Chiaverini and Sciavicco, 1993] as our force/position control technique of choice because it "offers some robustness with respect to the uncertainties affecting the knowledge of the environment." The parallel approach was implemented using position-based force control (PBFC), which is also know as inner position loop force control. The issues of PBFC have been studied by many researchers including Maples and Becker [1986], Reynolds *et al.* [1993], Pelletier and Doyon [1994], Heinrichs and Sepehri [1999], and Ferretti *et al.* [2000]. PBFC implementation was chosen due to the fact that TBRR was a *position-level inverse kinematics* method and thus a successful integration with PBFC was probably more likely than with direct force control implementation. Figure 8-7 shows the block diagram of how TBRR, parallel control, and PBFC came together in our implementation.

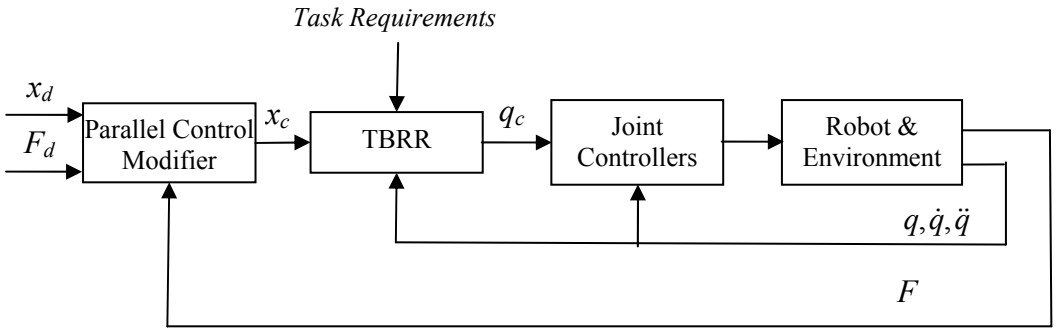


Figure 8-7: Integration of TBRR with parallel control implemented by PBFC (repeated from Section 7.3.2)

In order to carry out the simulations of force control with TBRR (as far as software implementation is concerned), we had to add a preliminary control framework and a dynamic simulator to OSCAR libraries. The control framework is currently being extended and improved by Rabindran and Tesar [2004].

In Section 7.5, the integrated task-based force control system was tested on a 10-DOF simulated robot in two different situations. In the first experiment, the robot was to track a circular path on a plane parallel to the X-Y plane while exerting force in the –Z direction (orthogonal to the plane). Here, the control system was successful in guiding the robot to track both position and force fairly accurately. The second experiment involved an unplanned contact situation in which the robot followed a circular path in free space. However, a wall was located in the robot path and prevented the robot from completing the path. In this type of unplanned contact situations, if only position control or hybrid position/force control were employed, it would usually result in excessive peak interaction force that could damage the robot itself and/or its surroundings. With the proposed system, the peak force was held to just 160 N compared with 2350 N that would have resulted with the pure position control.

- Several existing motion/force control techniques and implementations were reviewed.
- Position-based force control implementation was chosen because TBRR is a position-level inverse kinematics.
- Parallel approach was chosen as our motion/force control law.
- Parallel control with PBFC was integrated into TBRR framework.
- Simulations showed a successful integration of force control with TBRR.
- Dynamic simulator and preliminary control library were created from this work and will eventually be added into OSCAR.

Table 8-8: Summary of force control integration.

8.1.7 Summary of Primary Contributions

The research presented in this report aims at promoting usability of robots with redundancy by simplifying the deployment process. It incorporates task concepts into the decision making and control framework, which raises the operator's role to a level he can better understand and relate to. Several components were developed and seamlessly integrated for the framework to work as a whole. The key contributions of this research are summarized in the following table.

- Identification of weaknesses of generally accepted practices in existing RRTs.
  - Process of selecting and combining performance criteria for a given task was difficult and confusing.
  - Weighted sum of criteria had some major flaws such as normalization, couplings and conflicts among criteria, etc.
  - Purely kinematics with dynamics and control usually unaddressed.
- Improvements on the Generate Options scheme
  - Greatly reduces the chances of algorithmic singularities.
  - Improves trajectory smoothness while maintaining optimality.
- Development of the Vector Expansion method
  - Provides quick and accurate estimations of robot capabilities.
  - Particularly suitable for operational purposes.
- Development of Task-Based Redundancy Resolution
  - Raises the user's role to the task level where it was more easily understood.
  - Invaluable information of performance reserves is readily available at no extra computation cost.
  - More effective and responsive to changes in task requirements than traditional RRTs.
- Learning of subjective parameters
  - Decreases the user's reliance on the tedious and time-consuming trial and error approach.
- Integration of force control with TBRR framework
  - Makes TBRR equally applicable to contact as well as non-contact tasks.

Table 8-9: Summary of primary contributions from this research.

8.2 Future Work

Despite the many advances this research has contributed to the field of decision making and control of redundant manipulators, there is certainly room for improvement. This section outlines recommendations and suggestions for future research. Specific suggestions for immediate research as a continuation of this research are summarized in Table 8-10 and recommendations for long-term research in Table 8-11. Some of these recommended actions are detailed in the following subsections.

- Experiment the task-based decision making and control framework on physical robots.
- Further evaluate TBRR in several different conditions (different robots and different tasks) to ensure its effectiveness under various conditions.
- Incorporate more realistic actuator models (performance maps, torque-speed curves, joint errors such as back lash, etc.) in order to obtain more accurate estimations of robot capabilities.
- Detailed analysis of how Jacobian decomposition affects robot capability estimations.
- Develop tool/task models for several common robotic tasks for better understanding of tool and task relations. This development would help supply more precise task requirement specifications to TBRR.
- Extend the parameter learning algorithm to other subjective parameters such as perturbation size and momentum term.
- Comprehensive literature review on machine learning and other artificial intelligence techniques and their applications to advanced robotics, especially for skill acquisition.
- Develop a high-performance force/motion control method or perhaps a collection of methods suitable for various robotic tasks for integration with TBRR.
- Investigate a systematic reduction of task space dimensions on an array of common robotic tasks. This will allow TBRR to be deployed on six or fewer DOF robots that represent the majority of robot population.
- Apply robot capability analysis, especially the VE method, to the design of robotic manipulators.

Table 8-10: Suggestions for near-term research actions.

- Investigate decision making in the task space to be used in online (real-time) path planning.
- Apply machine learning and artificial intelligence technologies to robotic skill acquisition.
- Integration of multiple sensors and study of sensor fusions to improve sensor feedback and to help compensate for incomplete or unavailable system and/or environment models.
- Investigate the effects of system parameters on performance reserves and how they can be used in condition-based maintenance and fault tolerance.

Table 8-11: Suggestions for long-term research actions.

8.2.1 Modeling

Modeling is crucial in every analytical tool and TBRR is no exception. Improvement in TBRR performance can only be achieved with improved modeling. Some of necessary modeling efforts that still require further development are listed here.

8.2.1.1 More Realistic Models of Joint Capabilities

In estimating the robot's capabilities, it was assumed that the actuators' torque limits were constant as we contended that we only calculated the static force capability. It would be more realistic to use the actuator's torque-speed curve to determine the torque limit at a given speed and use that information to estimate robot capabilities. A thorough study on measurements of actuator performance using a nonlinear test bed by UTRRG's Yoo and Tesar [2004] is almost completed. An example of performance maps that can be expected from this work is shown in Figure 8-8. Here, the actuator efficiency is measured and plotted against the motor speed and current. With a complete performance envelope of actuators, we should

be able to map any performance measures from the joint space to the task space as desired.

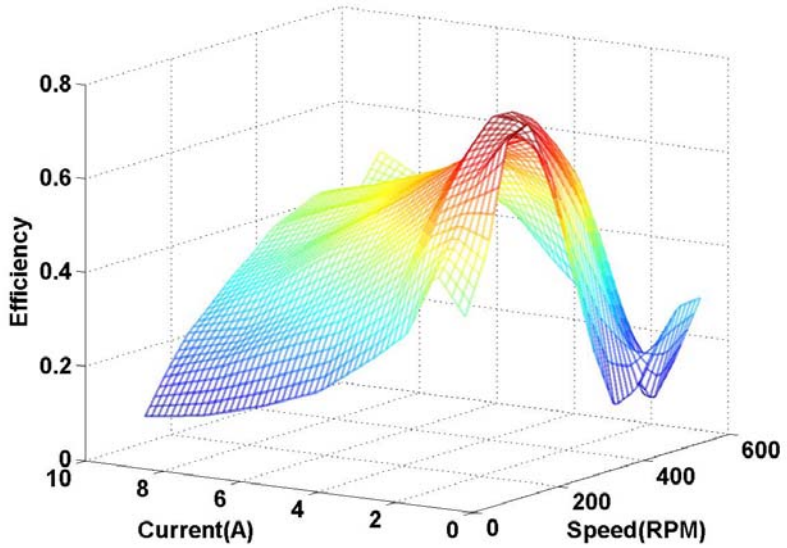


Figure 8-8: Actuator efficiency plot [Yoo and Tesar, 2004].

8.2.1.2 Metrology

The mathematical foundations of manipulator models, inverse kinematics schemes, performance criteria, and robot capability estimations rely heavily on the existence of accurate system parameters such as D-H parameters, inertia properties, compliances, etc. Therefore it is critical that such parameters be obtained accurately *as-built*, not as-designed. UTRRG is currently developing an indoor Global Positioning System (iGPS) metrology system with measurement accuracy of 0.002 inch [Kang *et al.*, 2004]. The main components of the iGPS metrology systems are the iGPS components (transmitters, receivers, control center, etc.) and a newly developed 3D probe. With this metrology system and the necessary kinematic and compliance analyses, we should be able to accurately measure geometric and compliance properties of robotic systems. Figure 8-9 shows the iGPS components and 3D probe and their application areas.

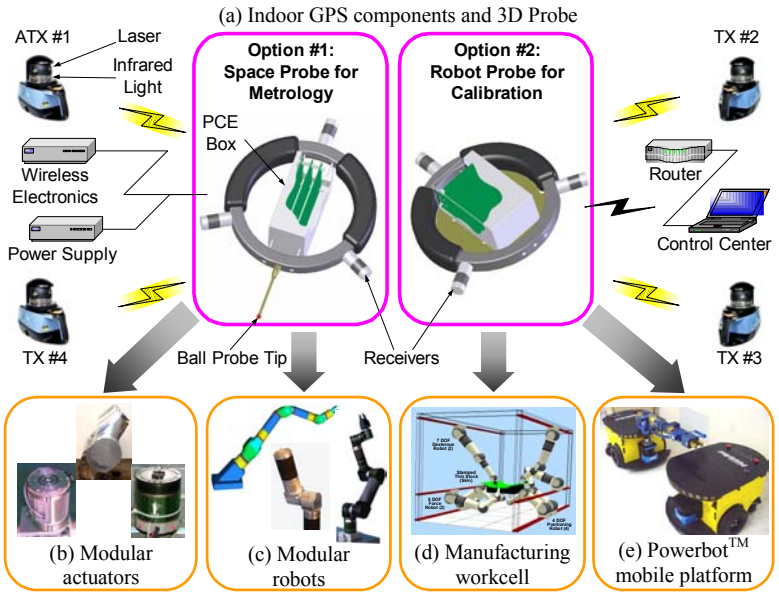


Figure 8-9: Indoor GPS metrology system with 3D Probe and application areas [Kang *et al*., 2004].

8.2.1.3 Tool/Task Modeling and Analysis

In the TBRR algorithm, it was assumed that task requirement specifications were given and the TBRR algorithm used those specifications in its determination of the optimal configuration. But how do we get these task requirement values? The answer to this question lies in the modeling and analysis of tasks and tools. Mathematical descriptions of a group of core robotic processes (drilling, grinding, deburring, sawing, peg-into-hole assembly, screw fastening, etc.) and their related tools should be analytically obtained. This subject is being investigated by Chang and Tesar [2004]. An example of tool modeling is shown in Figure 8-10. Here, wrenches associated with a drilling process are analyzed. Figure 8-11 depicts a performance map of a horizontal force with respect to feed-rate and deflections obtained from this analysis and illustrates the kind of results we should expect to gain when the study of tools is complete.

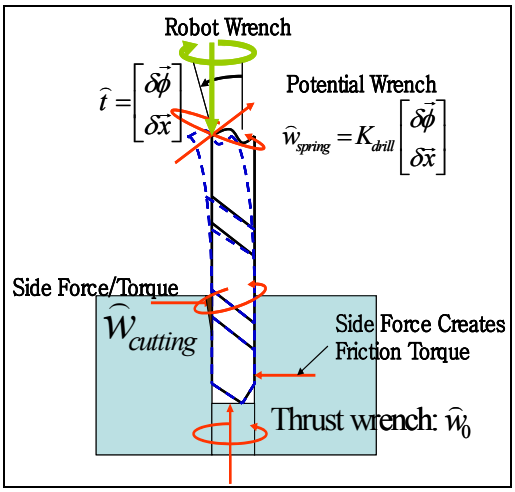


Figure 8-10: Wrenches in drilling process [Chang and Tesar, 2004].

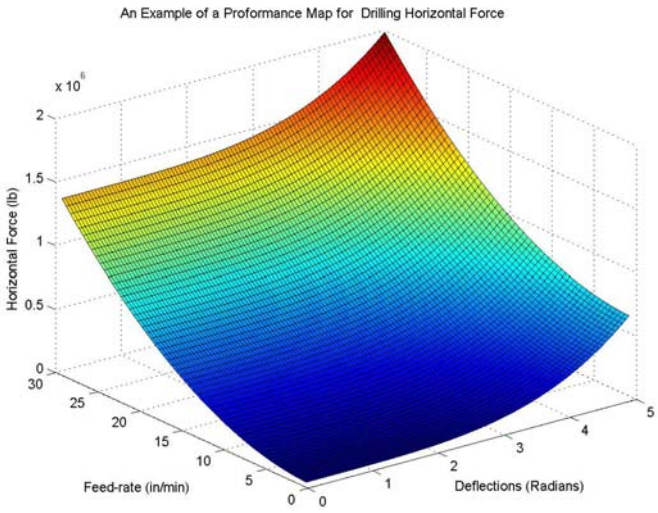


Figure 8-11: Performance Map for a Horizontal Force in Drilling [Chang and Tesar, 2004].

Based upon the tool/task models, when the robot picks up a tool, the system should automatically generate relevant task requirement specifications for that tool and feed them to the TBRR algorithm so that the robot can reconfigure itself accordingly. Moreover, other tool constraints should be automatically imposed. For example, when a circular saw is on, the robot should only be allowed to move

in the plane containing the saw blade. Movements outside this plane can cause the saw blade to bind.

8.2.2 Decision Making

Over the past few decades, there have been countless research efforts devoted to decision making of redundant manipulators. Most of these works have concentrated on the manipulator itself instead of the task being performed, as evident by a large number of performance criteria whose meanings to the task are questionable at best. It is our belief that going forward the focus of the decision-making problem in robotics should be shifted towards a higher level of task.

8.2.2.1 Decision Making in Task Space

The premise of most decision-making schemes on redundant systems is that the EEF position is given either by predefined trajectory, through input devices (in teleoperation), or some simple path generation algorithms. The framework developed in this report then determines the best configuration for the given EEF position. Nevertheless, the predefined EEF trajectory sent to the TBRR algorithm may be not be appropriate or outdated as the environment changes. Therefore, it is sometimes necessary that the path be adapted online during execution. Look at how humans move their hands for example. We may initially plan a high-speed path to move our hand to the destination as quickly as possible. During the execution, however, a moving object suddenly approaches and impedes the path. Naturally, the first human reaction is to slow down the hand movement. If that is still insufficient, the next reaction is to alter the path originally planned. This level of sophistication is what we should strive to achieve in robots too.

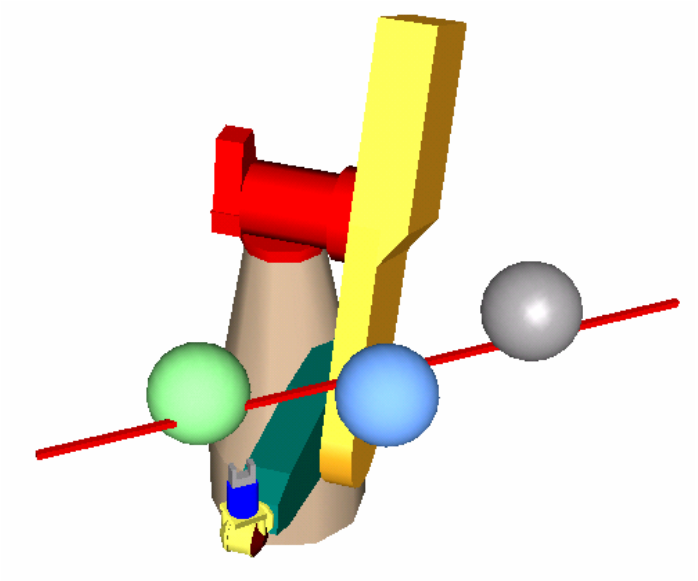


Figure 8-12: Online path adjustment for a PUMA robot [Tisius *et al.*, 2004].

UTRRG has recently begun investigating online path adjustment. In a preliminary work, a performance criterion in the EFF space was created to measure the distance from the EFF to an object [Tisius *et al.*, 2004]. This distance information can be used to repel the EEF from or attract the EEF to the object, depending on whether the object is an obstacle or a destination. Figure 8-12 depicts a PUMA robot adapts its path to avoid three obstacles (spheres) using the defined EEF criterion.

With the robot capability analysis presented in this report, we are already equipped with a tool that helps determine if the robot is able to execute the current task plan. When the system detects that the robot's performance reserves (e.g. for force, velocity, or acceleration) are being depleted, it can alert the user or an online path generator so that a new, executable task plan can be produced.

8.2.2.2 Performance Reserves

One of the benefits of using TBRR with the VE method is that the real-time information on performance reserves is readily available with no extra

computational cost. Tracking the values of performance reserves, which are defined as the differences between robot capabilities and task requirement specifications, can be very useful not only in decision making but also in condition-based maintenance and fault tolerance. The effects of system parameters such as buffer sizes on performance reserves should be fully investigated. In addition, action must be taken once the reserves have been determined to disappear. Robot capability estimations should serve as a tool to determine which subsystem is most critical to this decision as to whether the subsystem should be replaced (by plug-and-play) or reconfigured (by fault-tolerant control scheme) to bring the performance reserves up to desired levels. A systematic way of determining what action to take for the best results should be explored.

8.2.2.3 Learning and Skill Acquisition

Humans acquire new skills by learning and practice. When they face a new task that they have never performed before, their initial productivity is understandably low. As they perform the task over and over again, they become more skilled at that task and the productivity begins to rise. Skill acquisition is paramount if we wish to advance the robot technology and enable the robots to perform a wider variety of tasks at a high level.

The first challenge is representations of tasks that are suited for skill acquisition and learning. Numerical representations of tasks such as desired position and force trajectories are appropriate for robot kinematics and control but far too low-level and detail-oriented to be of any use for learning. Tasks must be parameterized and captured so as to describe their essence. For example, in optimizing a walk by the Sony Aibo robot, Kohl and Stone [2004] describe the walk by 12 parameters and use a learning approach to obtain the parameter values that yield the fastest walk for the robot for that particular parameterization.

Skill acquisition goes beyond just parameterization of tasks to include control actions as well. For instance, when humans perform a peg-in-hole insertion[27], we often employ "compliant" motion control in which we react to a force feedback that occurs during insertion. We must *learn* how to react properly in order to successfully insert the peg into the hole. Proper reaction depends on several factors including the alignment of peg and hole, the approach angle, tolerance, material properties of peg and hole, etc. Gullapalli *et al.* [1994] applied reinforcement learning to skill acquisition of peg-in-hole insertion by a Zebra Zero robot, which is a small 6-DOF manipulator. Basically, the reactive control strategy was learned by "active generation of compliant behavior using nonlinear mapping from sensed positions and forces to position commands. The controller learns this mapping through repeated attempts at peg insertion." Even though they were able to learn a successful reactive control strategy with this approach, it took approximately 800 trials before the control became skilled. One reason might have been that the learning algorithm had no prior knowledge of the task at hand. If a good peg-in-hole task model (for example from task modeling and analysis discussed in Section 8.2.1.3) was available, the learning algorithm could use it as a guide in its search for a better controller, which could save significant time in the training period and could result in a better controller as well.

8.2.3 Actuation and Control

The robot is only as good as its components and actuators are perhaps the most important ones. Continued efforts to produce lighter, more compact, more powerful, yet forgiving actuators are needed. Distributed Macro-Mini (DM^2) actuation approach has been proposed to simultaneously achieve high performance

---

[27] Peg-in-hole insertion is particularly important to industrial robotics since around 33% of assembly tasks are peg-in-hole insertions [Gullapalli *et al.*, 1994]

and safety requirements [Zinn *et al.*, 2004]. The DM^2 approach employs two actuators in *parallel* with the large actuator acting at low frequency and the small one at high frequency. These two actuators are *distributed* in the sense that the small one is located at the joint while the large one at the base. The parallel and distributed actuation is shown in the diagram in Figure 8-13.

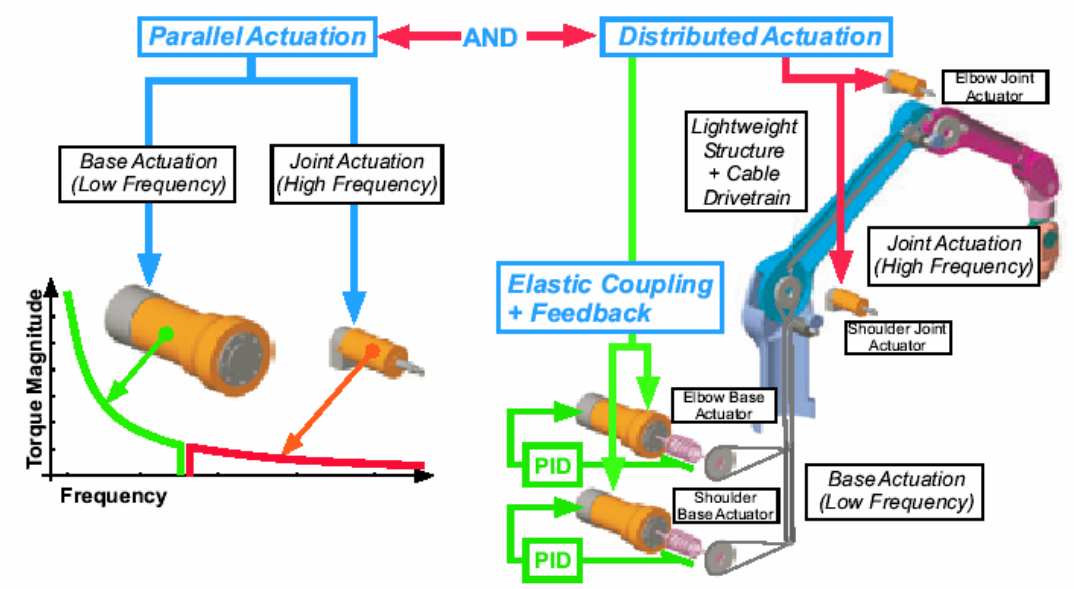


Figure 8-13: Distributed Macro-Mini Actuation Approach [Zinn *et al.*, 2004].

Focusing primarily on performance and control, UTRRG has conceptually designed a dual force/motion actuator. This actuator has two prime movers connected through a cleverly-designed hypocyclic gear train that combines the two inputs and gives out one output. Our premise is that this actuator will be able to fulfill both force and motion demands. While in motion control mode, the velocity prime mover is responsible for the motion control and the torque prime mover rejects disturbance. In force control, the opposite is true. This concept is being investigated [Rabindran and Tesar, 2004] and if it shows promise, a new force/motion control algorithm will be developed based on this special actuator. This new force/motion control algorithm would potentially be integrated with TBRR in the future.

8.2.4 Sensor Integration

Sensing is an integral part in controlling any robotic system. The more sensors are incorporated, the more information we can gather and the more sophisticated the system becomes. The system needs to sift through all the data and selectively include, discard, or even fuse some of these data. Sensors can be at actuator or system level. Actuator-level sensors provide information regarding the actuators themselves such as joint positions, joint torques, motor currents, etc. The actuator sensory information is utilized not only in actuator controller but also in other activities such as condition-based maintenance [Vasquez and Tesar, 2000] [Hvass and Tesar, 2004]. System-level sensors – such as force/torque sensors, vision systems, proximity sensors, range finders, and tactile sensors are essential for several robot control tasks including force/motion control, collision avoidance, grasping, navigation, etc.

In decision making, computation of several components relies on sensory information. For example, most performance criteria and robot capability estimations are functions of joint positions. In obstacle avoidance, minimum distances of complex-shape objects can be difficult to calculate. Proximity sensors, range finders, or sometimes vision systems, however, can replace these minimum distance calculations. The inverse kinematics part of decision making often assumes that the desired trajectory is known *a priori*. In many real tasks, however, this may not be the case. Or even if the trajectory is given, it may not be consistent with the real environment because of some errors in modeling of the environment. In these cases, sensory information can be used in real-time to augment the environmental models; or in the absence of these models, the motion can even be planned based entirely on sensory information. For example, Xiao *et al.* [1998] utilized sensor fusion of information from encoders, a force/torque sensor, and a camera in hybrid position/force control on an unknown constrained surface. These types of sensor integration advancements are needed if complex decision making

systems are to be deployed in real systems. Another issue is sensor noise. Since every sensor is bound to have some degree of noise, robustness against noise must be considered in the future work on decision making systems.

8.2.5 Applications

TBRR was developed for redundant manipulators, which are mostly deployed in advanced applications such as space robotics, telerobotics, or D&D types of applications. Redundant robots, however, represent only a fraction of robot population. Most commercially available industrial robots have six degrees of freedom or fewer and are being utilized in a traditional fashion. Nonetheless, these industrial robots sometimes perform tasks that require fewer than the degrees of freedom they possess and consequently can be viewed as redundant robots. Therefore, in order for TBRR to be more widely adopted, an emphasis should be placed on applying TBRR to improve performance of commercial robots.

In Chapter 5, we showed that, on a 6-DOF PUMA robot, when the orientation was not restricted, TBRR could be used to improve its task performance. However, this demonstration was simplified to make a point. In practice, rarely if ever are all three rotational degrees of freedom allowed to be free. More realistically, one or at most two degrees of freedom may not be used at a given time. One example of this is a robotic drilling task. Obviously the roll motion (the rotation whose axis coincides with the drill bit) of the robot's EEF does not affect the task and therefore needs not be specified. A thorough investigation should be conducted to obtain a systematic reduction of the task space dimensions on a variety of common tasks. TBRR might need to be 'tweaked' to perform more effectively and efficiently on these reduced dimensional task space types of applications.

Appendix A

PROPERTIES OF SELECTED MANIPULATORS

This appendix describes the kinematic, dynamic, and compliance properties as well as constraints of the computer-simulated manipulators used in this report.

A.1 4R Planar Manipulator

DH Parameters

```
This is a DH parameter file for a 4R Planar robot.  The notation is
based on the "Introduction to Robotics, Mechanics and Controls book
by John Craig.  All units are SI.

  Alpha(i-1)   a(i-1)        d(i)          theta(i)
  0.0          0.0           0.0           var        # link 0-1
  0.0          0.3           0.0           var        # link 1-2
  0.0          0.24          0.0           var        # link 2-3
  0.0          0.1           0.0           var        # link 3-4
Note: The tool point is (0.08,0,0).
```

Centers of Mass

```
This is a link Center of gravity data file for a 4R Planar robot.
Each row specifies the coordinates of the C.G. of the link (in its
local frame) and its mass.  The first row is for link 1, the second
for link 2, etc.  All units are SI (meters for length and kg for
mass).

Coord: X       Y       Z       mass
      0.20    0.0     0.0      7.5     # link 1
      0.15    0.0     0.0      5.0     # link 2
      0.08    0.0     0.0      2.0     # link 3
      0.06    0.0     0.0      1.0     # link 4
```

Link Inertias

```
This is a link Inertia Tensor data file for a 4R Planar robot.
Each three rows specify the inertia tensor of a link.  These
inertias are about the links' centers of mass.  The first three
rows are for link 1, the second three rows for link 2, etc.  All
units are SI.

        0.0     0.0     0.0     # link 1
        0.0     0.0     0.0
        0.0     0.0     0.215
```

```
0.0     0.0     0.0     # link 2
0.0     0.0     0.0
0.0     0.0     0.165

0.0     0.0     0.0     # link 3
0.0     0.0     0.0
0.0     0.0     0.075

0.0     0.0     0.0     # link 4
0.0     0.0     0.0
0.0     0.0     0.013
```

Joint Compliance Matrix

This is a joint compliance matrix data file for a 4R Planar robot. The units are rad/(N-m).

```
0.00001    0.0        0.0        0.0
0.0        0.00001    0.0        0.0
0.0        0.0        0.00001    0.0
0.0        0.0        0.0        0.00001
```

Joint Limits

This file contains the joint limits for the 4-DOF Planar robot. Units are in degrees.

```
Lower    Upper
-180     180     #Joint 1
-180     180     #Joint 2
-180     180     #Joint 3
-180     180     #Joint 4
```

A.2 10-DOF Fault Tolerant Manipulator

DH Parameters

This is a DH parameter file for a 10 DOF robot. The notation is based on the "Introduction to Robotics, Mechanics and Controls" book by John Craig. All units are SI.

```
  Alpha(i-1)    a(i-1)       d(i)       theta(i)
  0.0           0.0          0.0        var          # link 0-1
  90.0          0.0          0.0        var          # link 1-2
 -90.0          0.0          1.0        var          # link 2-3
  90.0          0.0          0.0        var          # link 3-4
 -90.0          0.0          0.9        var          # link 4-5
  90.0          0.0          0.0        var          # link 5-6
 -90.0          0.0          0.322      var          # link 6-7
  90.0          0.0          0.0        var          # link 7-8
 -90.0          0.0          0.0        var          # link 8-9
 -90.0          0.0          0.0        var          # link 9-10
```

Note: The tool point depends on the tool at hand.

Centers of Mass

```
This is a link Center of gravity data file for a 10 DOF robot.
Each row specifies the coordinates of the C.G. of the link (in its
local frame) and its mass.  The first row is for link 1, the second
for link 2, etc.  All units are SI (meters for length and kg for
mass).

Coord:  X        Y        Z         mass
       0.00     0.00    -0.4064    3.1751       # link 1
       0.00     0.00     0.00      1.7182       # link 2
       0.00     0.00     0.4826    0.9013       # link 3
       0.00     0.00     0.00      0.9013       # link 4
       0.00     0.00     0.4318    0.6055       # link 5
       0.00     0.00     0.00      0.2885       # link 6
       0.00     0.00     0.1524    0.2451       # link 7
       0.00     0.00     0.00      0.1127       # link 8
       0.00     0.00     0.00      0.0577       # link 9
       0.00     0.00     0.1778    0.0648       # link 10
```

Link Inertias

```
This is a link Inertia Tensor data file for a 10 DOF robot
Each three rows specify the inertia tensor of a link
The first three rows are for link 1, the second three rows for link
2, etc.  All units are SI.

     0.122706     0.0          0.0          #Inertia Tensor link 1
     0.0          0.122706     0.0
     0.0          0.0          0.0530564

     0.055035     0.0          0.0          #Inertia Tensor link 2
     0.0          0.055035     0.0
     0.0          0.0          0.018144

     0.008124     0.0          0.0          #Inertia Tensor link 3
     0.0          0.008124     0.0
     0.0          0.0          0.00572413

     0.008124     0.0          0.0          #Inertia Tensor link 4
     0.0          0.008124     0.0
     0.0          0.0          0.00572413

     0.002546007  0.0          0.0          #Inertia Tensor link 5
     0.0          0.002546007  0.0
     0.0          0.0          0.002747347

     0.002977     0.0          0.0          #Inertia Tensor link 6
     0.0          0.002977     0.0
     0.0          0.0          0.001141
```

```
0.0005294    0.0          0.0          #Inertia Tensor link 7
0.0          0.0005294    0.0
0.0          0.0          0.0004

0.001053     0.0          0.0          #Inertia Tensor link 8
0.0          0.001053     0.0
0.0          0.0          0.00023

0.000106     0.0          0.0          #Inertia Tensor link 9
0.0          0.000106     0.0
0.0          0.0          0.000056773

0.00009745   0.0          0.0         #Inertia Tensor link 10
0.0          0.00009745   0.0
0.0          0.0          0.000083608
```

Joint Compliance Matrix

```
This is a joint compliance matrix data file for the 10 DOF robot.
The units are rad/(N-m).

  0.0000257   0     0     0     0     0     0     0     0     0
  0     0.0000257   0     0     0     0     0     0     0     0
  0     0     0.0000257   0     0     0     0     0     0     0
  0     0     0     0.0000257   0     0     0     0     0     0
  0     0     0     0     0.0000481   0     0     0     0     0
  0     0     0     0     0     0.000197    0     0     0     0
  0     0     0     0     0     0     0.000197    0     0     0
  0     0     0     0     0     0     0     0.000385    0     0
  0     0     0     0     0     0     0     0     0.000385    0
  0     0     0     0     0     0     0     0     0    0.000385
```

Joint Limits

```
This file contains the joint limits for the 10-DOF robot.
Units are in degrees.

  Lower     Upper
  -270       270      #Joint 1
  -120       120      #Joint 2
  -270       270      #Joint 3
  -150       150      #Joint 4
  -270       270      #Joint 5
  -150       150      #Joint 6
  -270       270      #Joint 7
  -160       20       #Joint 8
   35        145      #Joint 9
  -360       360      #Joint 10
```

Joint Torque Limits
This file contains the joint torque limits for the 10-DOF robot. Units are in N-m.

```
  904      #Joint 1
  904      #Joint 2
  508      #Joint 3
  283      #Joint 4
  283      #Joint 5
  158      #Joint 6
  68       #Joint 7
  17       #Joint 8
  17       #Joint 9
  17       #Joint 10
```

Joint Speed Limits
All joint speed limits for the 10-DOF robot are 60 degrees/second.

Joint Encoder Resolutions
All joint encoder resolutions for the 10-DOF robot are 0.05 degrees.

A.3 Mitsubishi PA10-7CE Manipulator

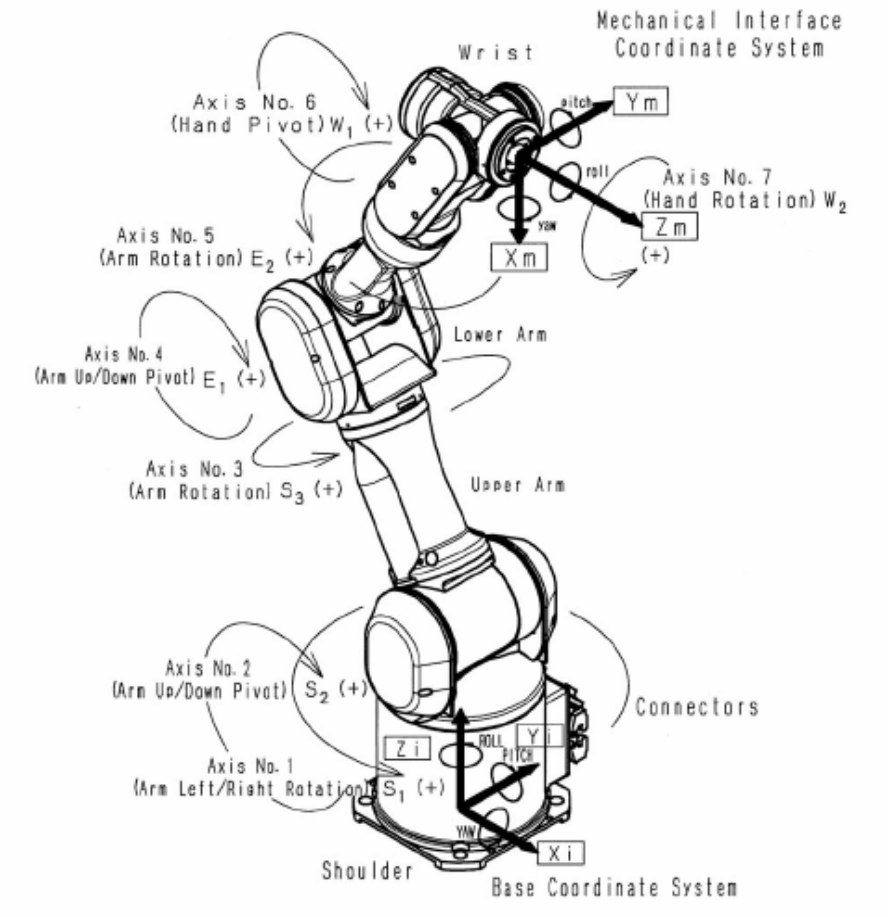


Figure A-1: Coordinate Systems of Mitsubishi PA10-7CE Robot [Mitsubishi Heavy Industries, Ltd.].

DH Parameters

This is a DH parameter file for the Mitsubishi robot. The notation is based on the "Introduction to Robotics, Mechanics and Controls" book by John Craig. All units are SI.

Alpha(i-1)	a(i-1)	d(i)	theta(i)	
0.0	0.0	0.0	var	# link 0-1
-90.0	0.0	0.0	var	# link 1-2
90.0	0.0	0.45	var	# link 2-3
-90.0	0.0	0.0	var	# link 3-4
90.0	0.0	0.48	var	# link 4-5
-90.0	0.0	0.0	var	# link 5-6
90.0	0.0	0.0	var	# link 6-7

Note: The tool point depends on the tool at hand.

Centers of Mass

This is a link Center of gravity data file for the Mitsubishi robot. Each row specifies the coordinates of the C.G. of the link (in its local frame) and its mass. The first row is for link 1, the second for link 2, etc. All units are SI (meters for length and kg for mass).

```
Coord:  X        Y        Z         mass
       0.00     0.00    -0.01       9.22        # link 1
       0.00    -0.2      0.0        4.51        # link 2
       0.00     0.00    -0.035      5.64        # link 3
       0.00    -0.115    0.0        2.04        # link 4
       0.00     0.00    -0.084      2.61        # link 5
       0.00    -0.042    0.0        2.07        # link 6
       0.00     0.00     0.022      1.05        # link 7
```

Link Inertias

This is a link Inertia Tensor data file for the Mitsubishi robot. Each three rows specify the inertia tensor of a link. The first three rows are for link 1, the second three rows for link 2, etc. All units are SI.

```
     0.122706     0.0          0.0          #Inertia Tensor link 1
     0.0          0.122706     0.0
     0.0          0.0          0.0530564

     0.055035     0.0          0.0          #Inertia Tensor link 2
     0.0          0.055035     0.0
     0.0          0.0          0.018144

     0.008124     0.0          0.0          #Inertia Tensor link 3
     0.0          0.008124     0.0
     0.0          0.0          0.00572413

     0.008124     0.0          0.0          #Inertia Tensor link 4
     0.0          0.008124     0.0
     0.0          0.0          0.00572413

     0.002546007  0.0          0.0          #Inertia Tensor link 5
     0.0          0.002546007  0.0
     0.0          0.0          0.002747347

     0.002977     0.0          0.0          #Inertia Tensor link 6
     0.0          0.002977     0.0
     0.0          0.0          0.001141

     0.0005294    0.0          0.0          #Inertia Tensor link 7
     0.0          0.0005294    0.0
     0.0          0.0          0.0004
```

Joint Compliance Matrix
This is a joint compliance matrix data file for the Mitsubishi robot. The units are rad/(N-m). The compliance values are obtained from Tsumugiwa *et al*. [2003]

```
0.000037  0       0     0     0     0     0
0     0.000037  0       0     0     0     0
0     0     0.0000588  0      0     0     0
0     0     0     0.0000909  0      0     0
0     0     0     0     0.000227  0       0
0     0     0     0     0     0.0005      0
0     0     0     0     0     0     0.000556
```

Joint Limits
This file contains the joint limits for the Mitsubishi robot.
Units are in degrees.

```
Lower     Upper
-180       180      #Joint 1
-97        97       #Joint 2
-180       180      #Joint 3
-143       143      #Joint 4
-270       270      #Joint 5
-180       180      #Joint 6
-270       270      #Joint 7
```

Joint Torque Limits
This file contains the joint torque limits for the Mitsubishi robot. Units are in N-m.

```
158      #Joint 1
158      #Joint 2
68       #Joint 3
68       #Joint 4
17       #Joint 5
17       #Joint 6
17       #Joint 7
```

Joint Speed Limits
This file contains the joint speed limits for the Mitsubishi robot.
Units are in degrees/second.

```
57       #Joint 1
57       #Joint 2
114      #Joint 3
114      #Joint 4
360      #Joint 5
360      #Joint 6
360      #Joint 7
```

Joint Encoder Resolutions
All joint encoder resolutions for the Mitsubishi robot are 0.05 degrees.

A.4 PUMA 760 Manipulator

DH Parameters
This is a DH parameter file for the Puma robot. The notation is based on the "Introduction to Robotics, Mechanics and Controls" book by John Craig. All units are SI.

```
  Alpha(i-1)    a(i-1)       d(i)       theta(i)
   0.0          0.0          0.0        var          # link 0-1
 -90.0          0.0          0.0        var          # link 1-2
   0.0          0.65         0.2        var          # link 2-3
 -90.0          0.0          0.69       var          # link 3-4
  90.0          0.0          0.0        var          # link 4-5
 -90.0          0.0          0.0        var          # link 5-6
```

Note: The tool point depends on the tool at hand.

Centers of Mass
This is a link Center of gravity data file for the PUMA robot. Each row specifies the coordinates of the C.G. of the link (in its local frame) and its mass. The first row is for link 1, the second for link 2, etc. All units are SI (meters for length and kg for mass).

```
Coord:  X        Y        Z         mass
       0.00     0.00    -0.4064     17.4        # link 1
       0.068    0.006   -0.016      17.4        # link 2
       0.00    -0.143    0.014      4.8         # link 3
       0.00    -0.115   -0.019      0.83        # link 4
       0.00     0.0      0.0        0.34        # link 5
       0.00     0.0      0.032      0.09        # link 6
```

Link Inertias
This is a link Inertia Tensor data file for the Puma robot. Each three rows specify the inertia tensor of a link. The first three rows are for link 1, the second three rows for link 2, etc. All units are SI.

```
      0.0         0.0         0.0         #Inertia Tensor link 1
      0.0         0.0         0.0
      0.0         0.0         0.35

      0.13        0.0         0.0         #Inertia Tensor link 2
      0.0         0.524       0.0
      0.0         0.0         0.539
```

```
0.066       0.0         0.0         #Inertia Tensor link 3
0.0         0.0125      0.0
0.0         0.0         0.086

0.0018      0.0         0.0         #Inertia Tensor link 4
0.0         0.0018      0.0
0.0         0.0         0.0013

0.0003      0.0         0.0         #Inertia Tensor link 5
0.0         0.0003      0.0
0.0         0.0         0.0004

0.00015     0.0         0.0         #Inertia Tensor link 6
0.0         0.00015     0.0
0.0         0.0         0.00004
```

Joint Compliance Matrix

This is a joint compliance matrix data file for the Puma robot.
The units are rad/(N-m).

```
  0.00000257  0     0     0     0     0
  0     0.00000257  0     0     0     0
  0     0     0.00000257  0     0     0
  0     0     0     0.0000481   0     0
  0     0     0     0     0.0000481   0
  0     0     0     0     0     0.0000481
```

Joint Limits

This file contains the joint limits for the Puma robot.
Units are in degrees.

```
  Lower     Upper
  -160       160      #Joint 1
  -110       110      #Joint 2
  -135       135      #Joint 3
  -266       266      #Joint 4
  -100       100      #Joint 5
  -300       300      #Joint 6
```

Joint Torque Limits

This file contains the joint torque limits for the Puma robot.
Units are in N-m.

```
  97.6       #Joint 1
  186.4      #Joint 2
  89.4       #Joint 3
  24.2       #Joint 4
  20.1       #Joint 5
  21.3       #Joint 6
```

Joint Speed Limits
All joint speed limits for the Puma robot are 60 degrees/second.

Joint Encoder Resolutions
All joint encoder resolutions for the Puma robot are 0.05 degrees.

REFERENCES

Anderson, R. and Spong, M.W., 1987, "Hybrid Impedance Control of Robotic Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 1073-1080.

Baillieul, J., Hoolerbach, J., and Brockett, R., 1984, "Programming and Control of Kinematically Redundant manipulators," *Proceedings of IEEE Conf. on Decision and Control*, pp. 768-774.

Berger, R.M. and ElMaraghy, H.A., 1992, "Feedback Linearization Control of Flexible Joint Robots," *Robotics & Computer-Integrated Manufacturing*, v. 9, n. 3, pp. 239-246.

Bevill, P.J. and Tesar, D., 1990, *Criteria Normalization to Support Decision Making in Intelligent Machines*, Master's Thesis, University of Texas at Austin.

Bowling, A. and Khatib, O., 1995, "Analysis of the Acceleration Characteristics of Non-Redundant Manipulators," *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 323-328.

Browning, G.T. and Tesar, D., 1996, *The Physical Significance of Kinematic and Dynamic Performance Criteria for Serial Redundant Manipulators*, Master's Thesis, University of Texas at Austin.

Carignan, C., 1991, "Trajectory Optimization for Kinematically Redundant Arms," *Journal of Robotic Systems*, v. 6, n. 5, pp. 221-248.

Cetin, M. and Tesar, D., 1999, *Performance Identification and Multi-Criteria Redundancy Resolution for Robotic Systems*, Ph.D. Dissertation, University of Texas at Austin.

Cetin, M., Lasdon, L., Kapoor, C., and Tesar, D., 1999, "Evaluation of Optimality in Two Classes of Local Redundancy Resolution Problems," *Proceedings of the 10th World Congress on the Theory of Machine and Mechanisms,* Finland, June 1999.

Chan, S.P. and Liaw, H.C., 1996, "Generalized Impedance Control of Robot for Assembly Tasks Requiring Compliant Manipulation," *IEEE Transactions on Industrial Electronics*, v. 43, n.4, pp. 453-461.

Chang, K. and Tesar, D., 2004, *Physical Modeling of Tools Necessary for Robot Manipulation*, Ph.D. Dissertation Proposal, University of Texas at Austin.

Chen, Y.C. and Walker, I.D., 1993, "A Consistent Null-Space Based Approach to Inverse Kinematics of Redundant Robots," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 374-381.

Cheng, F.-T., Chen, T.-H., and Sun, Y.-Y., 1992, "Efficient Algorithm for Resolving Manipulator Redundancy – The Compact QP Method," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 508-513.

Cheng, F.-T., Wang, W.-M., and Kung, F.-C., 1996, "Priority Considerations for Multiple Goals of Redundant Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 264-269.

Cheng, F.-T., Shih, M.-S., Kung, F.-C., and Sun, Y.-Y., 1997, "The Improved Parallel Scheme for Multiple-Goal Priority Considerations of Redundant Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 2409-2414.

Chiacchio, P., Bouffard-Vercelli, Y., and Pierrot, F., 1997, "Force Polytope and Force Ellipsoid for Redundant Manipulators," *Journal of Robotic Systems*, v. 14, no. 8, pp. 613-620.

Chiacchio, P. and Concilio, M., 1998, "The Dynamic Manipulability Ellipsoid for Redundant Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pp. 95-100.

Chiaverini, S. and Sciavicco, L, 1993, "The Parallel Approach to Force/Position Control of Robotic Manipulators," *IEEE Transactions on Robotics and Automation*, v. 9, no. 4.

Chiu, S., 1987, "Control of Redundant Manipulators for Task Compatibility," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp. 1718-1724.

Chiu, S., 1988, "Kinematic Characterization of Manipulators: An Approach to Defining Optimality," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, pp. 828-833.

Choi, M.H., 1999, "Redundancy Resolution by Minimization of Joint Disturbance Torque for Independent Joint Controlled Manipulators," *Proceedings of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Atlanta, GA, pp. 392-397.

Cocca, C. and Tesar, D., 1999, *Failure Recovery in Redundant Serial Manipulators*, Ph.D. Dissertation, University of Texas at Austin.

Craig, J.J., 1989, Introduction to Robotics, Addison Wesley, Massachusetts.

Denavit, J. and Hartenberg, R., 1955, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, v. 22, pp. 139-144.

Doty, K.L., Melchiorri, C., Schwartz, E.M., and Bonivento, C., 1995, "Robot Manipulability," *IEEE Transactions on Robotics and Automation*, v. 11, n. 3, pp. 462-468.

Duffy, J., 1990, "The Fallacy of Modern Hybrid Control Theory that is Based on "Orthogonal Complements" of Twist and Wrench Spaces," *Journal of Robotic Systems*, v. 7, no. 2, pp. 139-144.

Eschenbach, P. and Tesar, D., 1969, "Optimization of Four-Bar Linkages Satisfying Four Generalized Coplanar Positions," *Journal of Engineering for Industry*, pp. 75-82.

Featherstone, R., 1987, *Robot Dynamics Algorithms*, Kluwer Academic Publishers.

Ferretti, G., Magnani, G.A., Rocco, P., Cecconello, F., and Rossetti, G., 2000, "Impedance Control for Industrial Robots," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 4027-4032.

Finotello, R., Grasso, T., Rossi, G., and Terribile, A., 1998, "Computation of Kinetostatic Performances of Robot Manipulators with Polytopes," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 3241-3246.

Freeman, R.A., and Tesar, D., 1988, "Dynamic Modeling of Serial and parallel Mechanisms/Robotic Systems," *Trends and Developments in Mechanisms, Machines, and Robotics, Proceedings 20th Biennial ASME Mechanisms Conference*, pp 7-27.

Gullapalli, V., Franklin, J.A., and Benbrahim, H., 1994, "Acquiring Robot Skills via Reinforcement Learning," *IEEE Control Systems Magazine*, v. 14, issue 1, pp. 13-24.

Gosselin, C.M., 1990, "Dexterity Indices for Planar and Spatial Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 650-655.

Hanson, M.L. and Tolson, R.H., 1995, "Using a Fuzzy Supervisor to Optimize Multiple Criteria in Redundant Robots," *Proceedings of the 10th IEEE Int. Symp. On Intelligent Control*, pp. 145-150.

Harden, T. and Tesar, D., 1997, *The Implementation of Artificial Potential Field Based Obstacle Avoidance for a Redundant Manipulator*, Master's Thesis, University of Texas at Austin.

Harden, T., Kapoor, C., and Tesar, D., 1999, "Experimental Evaluation of a Criteria-Based Obstacle Avoidance Scheme," *Proceedings of ASME Design Engineering Technical Conferences*, Las Vegas, NV.

Hernandez, E. and Tesar, D., 1996, *Compliance Modeling for General Manipulator Structures*, Ph.D. Dissertation, University of Texas at Austin.

Hill, B. and Tesar, D., 1997, *Design of Mechanical Properties for Serial Manipulators*, Ph.D. Dissertation, University of Texas at Austin.

Hogan, N., 1985, "Impedance Control: An Approach to Manipulation," *Journal of Dynamic Systems, Measurement, and Control*, v. 107, pp. 1-24.

Hollerbach, J.M. and Suh, K.C., 1985, "Redundancy Resolution of Manipulators through Torque Optimization," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 1016-1021.

Hooper, R. and Tesar, D., 1994, *Multicriteria Inverse Kinematics for General Serial Robots*, Ph.D. Dissertation, University of Texas at Austin.

Hvass, P.B. and Tesar, D., 2004, *Condition Based Maintenance for Intelligent Electromechanical Actuators*, Master's Thesis, University of Texas at Austin.

Hwang, Y.-S., Lee, J., and Hsia, T.C., 2000, "A Recursive Dimension-Growing Method for Computing Robotic Manipulability Polytope," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 2569-2574.

Kang, H.-J. and Freeman, R.A., 1992, "Joint Torque Optimization of Redundant Manipulators via the Null Space Damping Method," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 520-524.

Kang, S.-H., Pryor, M., and Tesar, D., "Kinematic Model and Metrology System for Modular Robot Calibration," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 2894-2899.

Kapoor, C. and Tesar, D., 1996, *A Reusable Operational Software Architecture for Advanced Robotics*, Ph.D. Dissertation, University of Texas at Austin.

Kapoor, C., Cetin, M., and Tesar, D., 1998, "Performance Based Redundancy Resolution with Multiple Criteria," *Proceedings of 1998 ASME Design Engineering Technical Conference*, September 13-16, Atlanta, Georgia.

Kelmar, L. and Khosla, P., 1988, "Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, pp. 663-668.

Khatib, O., 1987, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, v. 3, pp 43-53.

Khatib, O., 1990, "Motion/Force Redundancy of Manipulators," *Proceedings of Japan-USA Symposium on Flexible Automation*, Kyoto, Japan, pp. 337-342.

Kim, H.S. and Choi, Y.J., 1999, "The Kinetostatic Capability Analysis of Robotic Manipulators," *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1241-1246.

Kohl, N. and Stone, P., 2004, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, pp. 2619-2624.

Lee, J., 1997, "A Study on the Manipulability Measures for Robot Manipulators," *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1458-1465.

Lee, S. and Lee, H.S., 1991, "Intelligent Control Manipulators Interacting with an Uncertain Environment Based on Generalized Impedance," *Proceedings of IEE Int. Symp. on Intelligent Control*, pp.61-66.

Li, Z. and Sastry, S., 1987, "Task Oriented Optimal Grasping by Multifingered Robot Hands," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp. 389-394.

Liegeois, A., 1977, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-7, n.12, pp. 868-871.

Liu, G.J. and Goldenberg, A.A., 1991, "Robust Hybrid Impedance Control of Robot Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 287-292.

Luenberger, D.G., 1984, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Co., 2nd Edition, Massachusetts.

Maples, J.A. and Becker, J.J., 1987, "Experiments in Force Control of Robotic Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 695-702.

March, P.S., Taylor, R.C., Kapoor, C., and Tesar, D., 2004, "Decision Making for Remote Robotic Operations," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 2764-2769.

Mason, M., 1981, "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-11, no. 6, pp. 418-432.

Mavroidis, C., Flanz, J., Dubowsky, S., Drouet, P., and Goitein, M., 1998, "High Performance Medical Robot Requirements and Accuracy Analysis," *Robotics and Computer-Integrated Manufacturing*, v. 14, pp. 329-338.

McGhee, S., Chan, T.F., Dubey, R.V., and Kress, R.L., 1994, "Probability-Based Weighting of Performance Criteria for a Redundant Manipulator," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, v. 3, pp. 1887-1894.

Mitsubishi Heavy Industries, Ltd., *Instruction Manual for Installation, Maintenance & Safety of PA10-7CE*.

Nakamura, Y., 1991, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley Publishing.

Nakamura, Y. and Hanafusa, H., 1986, "Inverse Kinematics Solution with Singularity Robustness for Robot Manipulator Control." *ASME Journal of Dynamic Systems, Measurements, and Control*, v. 108, n. 3, pp. 163-171

Nakamura, Y., Hanafusa, H., and Yoshikawa, T., 1987, "Task Priority Based Redundancy Control of Robot Manipulators," *Int. Journal of Robotics Research*, v. 6, n. 2, pp. 3-15.

Nemec, B., 1997, "Pseudoinverses and Null Space Velocity Controller," *Proceedings of IEEE Int. Conf. on Intelligent Engineering Systems*, pp. 107-111.

Nemec, B. and Zlajpah, L., 1998, "Experiments on Impedance Control of Redundant Manipulators," *Proceedings of IEEE Int. Symp. on Industrial Electronics*, pp. 134-138.

Newman, W.S. and Dohring, M.E., 1991, "Augmented Impedance Control of Kinematically Redundant Manipulators," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 30-35.

Nedungadi, A. and Kazerouinian, K., 1988, "A Local Solution with Global Characteristics for the Joint Torques Optimization of a Redundant Manipulator," *Journal of Robotic Systems*, v. 6, no. 5, pp. 631-654.

Oh, Y., Chung, W., and Youm, Y., 1998, "Extended Impedance Control of Redundant Manipulators Based on Weighted Decomposition of Joint Space," *Journal of Robotic Systems*, v. 15, n. 5, pp. 231-258.

Pelletier, M. and Doyon, M., 1994, "On the Implementation and Performance of Impedance Control on Position Controlled Robots," *Proceedings of IEEE*, pp. 1228-1233.

Peng, Z.-X. and Adachi, N., 1993, "Compliant Motion Control of Kinematically Redundant Manipulators," *IEEE Trans. on Robotics and Automation*, v. 9, n. 6, pp. 831-837.

Perry, B.R. and Tesar, D., 1995, *The Development of Distance Functions and Their Higher-Order Properties for Artificial Potential Field-Based Obstacle Avoidance*, Master's Thesis, University of Texas at Austin.

Pholsiri, C., Rabindran, D., Pryor, M., and Kapoor, C., 2003, "Extended Generalized Impedance Control for Redundant Manipulators," Accepted to *Proceedings of IEEE Conf. on Decision and Control*, Hawaii, Dec. 9-12.

Pholsiri, C., Kapoor, C., and Tesar, D., 2002, "Singularity Robust Inverse Kinematics for Teleoperation," *Proceedings of Spectrum 2002 Conf. on Nuclear and Hazardous Waste Management*, Reno, NV.

Pryor, M. and Tesar, D., 1999, *Complex Task Completion with Redundant Serial Manipulators*, Master's Thesis, University of Texas at Austin.

Pryor, M. and Tesar, D., 2002, *Task-Based Resource Allocation for Improving the Reusability of Redundant Manipulators*, Ph.D. Dissertation, University of Texas at Austin.

Rabindran, D. and Tesar, D., 2004, *Preliminary Studies in Force/Motion Control of Intelligent Mechanical Systems*, Master's Thesis, University of Texas at Austin.

Raibert, M. and Craig, J., 1981, "Hybrid Position/Force Control of Manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, v. 102, n. 2, pp. 126-133.

Reynolds, D.L., Carroll, J.J., Kriikku, E.M., and Dawson, D.M., 1993, "Modified Impedance Control for Robotic Saw Cutting: Simulation and Implementation in Three Degrees of Freedom," *Proceedings of the 5th Topical Meeting on Robotics and Remote Systems*, pp. 251-258.

Seraji, H., 1989, "Configuration Control of Redundant Manipulators: Theory and Implementation," *IEEE Transactions on Robotics and Automation*, v. 5, n. 4, pp. 472-490.

Seraji, H. and Colbaugh, R., 1990, "Singularity-Robustness and Task-Prioritization in Configuration Control of Redundant Robots," *Proceedings of IEEE Conf. on Decision and Control*, Honolulu, HI, pp. 3089-3095.

Shadpey, F., Patel, R.V., Balafoutis, C., and Tessier, C., 1995, "Compliant Motion Control and Redundancy Resolution for Kinematically Redundant Manipulators," *Proceedings of American Control Conference*, Seattle, Washington, pp. 392-396.

Siciliano, B. and Slotine, J.-J. E., 1991, "A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems," *Proceedings of Int. Conf. on Advanced Robotics*, Pisa, Italy, pp. 1211-1216.

Smith, D. and Lasdon, L., 1992, "Solving Large Sparse Nonlinear Programs using GRG," *ORSA Journal on Computing*, v.4, n.1.

Surdilovic, D. and Kirchhof, J., 1996, "A New Position Based Force/Impedance Control for Industrial Robots," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN, pp. 629-634.

Thomas, M. and Tesar, D., 1982, "Dynamic Modeling of Serial Manipulator Arms," *Journal of Dynamic Systems, Measurement, and Control,* v. 104, pp. 218-228.

Thomas, M., Yuan-Chou, H.C., and Tesar, C., 1985, "Optimal Actuator Sizing for Robotic Manipulators Based on Local Dynamic Criteria," *Journal of Mechanisms, Transmission, and Automation in Design*, v. 107, pp. 163-169.

Tisius Jr., M., Kapoor, C. and Tesar, D., 2004, *An Empirical Approach to Performance Criteria and Redundancy Resolution*, Master's Thesis, University of Texas at Austin.

Tsumugiwa, T., Yokogawa, R., and Hara, K., 2003, "Measurement Method for Compliance of Vertical-Multi-Articulated Robot – Application to 7-DOF Robot PA-10," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, pp. 2741-2746.

Van Doren, M. and Tesar, D., 1992, *Criteria Development to Support Decision Making Software for Modular, Reconfigurable Robotic Manipulators*, Master's Thesis, University of Texas at Austin.

Vasquez Arvallo, A. and Tesar, D., 2000, *Condition-Based Maintenance of Actuator Systems Using a Model-Based Approach*, Ph.D. Dissertation, University of Texas at Austin.

Volpe, R. and Khosla, P., 1993, "A Theoretical and Experimental Investigation of Explicit Force Control Strategies for Manipulators," *IEEE Trans. on Automatic Control*, v.38, n. 11, pp. 1634-1650.

Volpe, R. and Khosla, P., 1995, "The Equivalence of Second-Order Impedance Control and Proportional Explicit Force Control," *Int. Journal of Robotics Research*, v. 14, n. 6, pp. 574-589.

Whitney, D.E., 1969, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems*, v. MMS-10, n. 2, pp. 47-53.

Xiao, D., Ghosh, B.K, Xi, N., and Tarn, T.J., 1998, "Intelligent Robotic Manipulation with Hybrid Position/Force Control in an Uncalibrated

Workspace," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pp. 1671-1676.

Yoo, J.G. and Tesar, D., 2004, *Actuator Performance Envelope Through Nonlinear Test Bed*, Ph.D. Dissertation, University of Texas at Austin.

Yoshikawa, T., 1984, "Manipulability of Robotic Mechanisms," *Int. Journal of Robotics Research*, v. 4, no. 2, pp. 3-9.

Yoshikawa, T., 1985, "Dynamic Manipulability of Robot Manipulators," *Journal of Robotic Systems*, v. 2, n. 1, pp. 113-124.

Yoshikawa, T., 1991, "Translational and Rotational Manipulability of Robotic Manipulators," *Proceedings of Conference on Industrial Electronics, Control and Instrumentation IECON*, pp. 1170-1175.

VITA

Chalongrath Pholsiri was born in Bangkok, Thailand on December 10, 1973, the son of Somboon Pholsiri and Pongpen Pholsiri. After completing his work at Suankularb Wittayalai, Bangkok, Thailand, in 1990, he entered Chulalongkorn University, Bangkok, Thailand, where he received the degree of Bachelor of Engineering in April 1994, majoring in Mechanical Engineering. During his time at Chulalongkorn University, he participated in an internship program at Toyota Motors (Thailand), Col Ltd. During the next 9 months, he was employed at Act Consultants, Co. Ltd. in Bangkok for as a designer for air conditioning and plumbing systems. He then came to the US to pursue advanced degrees. In December 1995, he completed a pre-MBA program at the University of Dallas. The University of Texas at Austin awarded Chalongrath the degree of Master of Science (Mechanical Engineering) in December 1997. He then joined Robotics Research Group as a Graduate Research Assistant and now is a doctoral candidate under the supervision of Dr. Delbert Tesar.

Permanent Address: 2203 Faldo Cv.
Round Rock, TX 78664

This dissertation was typed by the author.